U N I V E R Z I T E T U N O V O M S A D U
TEHNIČKI FAKULTET "MIHAJLO PUPIN" ZRENJANIN

Mr BILJANA RADULOVIĆ

PROJEKTOVANJE BAZA PODATAKA U OBLASTI OBRAZOVNOG RAČUNARSKOG SOFTVERA

DOKTORSKA DISERTACIJA

ZRENJANIN, 1997.

UNIVERZITET U NOVOM SADU
TEHNIČKI FAKULTET "MIHAJLO PUPIN" U ZRENJANINU

KLJUČNA DOKUMENTACIJSKA INFORMACIJA

Redni broj
RBR

Identifikacioni broj
IBR

Tip dokumentacije
TD: Monografska publikacija

Tip zapisa
TZ: Tekstualni štampani materijal

Vrsta rada
VR: Doktorska disertacija

Autor
AU: Mr Biljana Radulović, asistent Tehničkog fakulteta "Mihajlo Pupin" u Zrenjaninu

Mentor/Komentor
MN: Dr Đorde Nadrljanski, redovni profesor

Naslov rada
NS: Projektovanje baza podataka u oblasti obrazovnog računarskog softvera

Jezik publikacije
JZ: Srpski

Jezik izvoda
JI: Srpski i engleski

Zemlja publikovanja
ZP: SR Jugoslavija

Uže geografsko područje
UGP: Vojvodina

Godina
GD: 1997.

Izdavač
IZ: Tehnički fakultet “Mihajlo Pupin”

Mesto i adresa
MS: 23000 Zrenjanin, Đure Đakovića bb

Fizički opis rada (broj poglavlja/strana/lit.citata/grafika/priloga)
FO: 9/194/89/25/5

Naučna oblast
OB: Informatika

Naučna disciplina
DI: Baze podataka

Predmetna odrednica/Ključne reči
PO: Projektovanje baza podataka, Deduktivne baze podataka, Otvoreni i zatvoreni svet, Obrazovni softver
UDK:

Čuva se
ČU: Biblioteka Tehničkog fakulteta

Važna napomena
VN: Nema

Izvod
IZ: U radu su definisane teorijske postavke povezivanja koncepata otvorenog i zatvorenog sveta u jedinstven sistem za rukovanje bazom podataka u režimu otvorenog, zatvorenog i otvorenog/zatvorenog sveta. Opisan je konkretan programski sistem BASELOG, koji je razvijen na datim teoretskim postavkama. Opisan je postupak projektovanja baza podataka u oblasti obrazovnog softvera, koji je zasnovan na BASELOG-sistemu.

Datum prihvatanja teme od strane NN Veća
DP:

Datum odbrane
DO:

Članovi Komisije (naučni stepen/ime i prezime/zvanje/fakultet)
KO:
Predsednik: _____
Član: _____
Član: _____

UNIVERSITY OF NOVI SAD
TECHNICAL FACULTY "MIHAJLO PUPIN" ZRENJANIN

KEY WORDS DOCUMENTATION

Accession number
ANO:

Identification number
INO:

Document type
DT: Monographic publication

Type record
TR: Textual material, printed

Content code
CC: Ph. D. Thesis

Author
AU: M. Sc. Biljana Radulović, assistant, Technical Faculty "Mihajlo Pupin", Zrenjanin

Mentor/Comentor
MN: Ph. D. Đorđe Nadrljanski, professor

Title
TI: Data base projecting in the field of education computer software

Language of text
LT: Serbian

Language of abstract
LA: Serbian and English

Country of publication
CP: Yugoslavia

Locality of publication
LP: Vojvodina

Publication year
PY: 1997.
Publisher

PB: Technical faculty “Mihajlo Pupin”

Publ. place
PP: 23000 Zrenjanin, Đure Đakovića bb

Physical description (number of chapters/pages/literature/graphics/additional lists)
PD: 9/194/89/25/5

Scientific field
SF: Informatics

Scientific discipline
SD: Data bases

Subject/Keywords
SKW: Data base projecting, Deductive data bases, Open and closed world, Education software
UC:

Holding data
HD: The Library of Technical Faculty

Note
N: None

Abstract
AB: In this work theoretical bases of connecting concept open and closed world in one data base management system which works through open, closed and open/closed world are defined. BASELOG-program system is descripted and developed on given theoretical bases. There is also descripted process of data base projecting in the field of education software which is based on BASELOG-system.

Accepted by the Scientific Board on
ASB:

Defended on
DE:

Thesis defend board (degree/name/surname/title/faculty)
DB:

President : _____
Member: _____
Member : _____

S A D R Ž A J

PREDGOVOR

Problemima projektovanja baza podataka u oblasti obrazovnog računarskog softvera nije poklanjana dovoljna pažnja, iako se smatra da će primena informatike u obrazovanju biti jedan od glavnih pravaca razvoja u budućnosti.

Projektovanje baza podataka u oblasti poslovnih informacionih sistema je bilo predmet mnogih istraživanja. Razvijene su metode za projektovanje baza podataka i one su implementirane ili u CASE alate za projektovanje baza podataka ili u softvere za rukovanje bazama podataka. Pri tom, korišćeni su neki od osnovnih modela podataka: relacioni, model entiteta i poveznika, objektno - orijentisani model.

Savremeni pravci razvoja baza podataka odnose se na povezivanje koncepata baza podataka sa mehanizmima zaključivanja - razvoj deduktivnih baza podataka, ili na razvoj objektno - orijentisanog modela podataka, što je posledica uočenih nedostataka relacionog modela podataka.

Osnovni cilj ovog istraživanja bio je povezati koncepte otvorenog i zatvorenog sveta u bazama podataka i izgraditi sistem, koji neće raditi samo na osnovu jednog principa, nego će imati mogućnost izbora režima rada. Ova mogućnost je naročito potrebna za rukovanje bazama znanja koje se koriste u projektovanju obrazovnog softvera, jer se pokazalo da koncept zatvorenog sveta nije podesan za primenu baza podataka u ovoj oblasti.

Prva tri poglavlja imaju za cilj definisanje osnovnih koncepata koji se koriste u ovom istraživanju. Glavni rezultati istraživanja se nalaze u četvrtom i petom poglavlju. Zaključak i rezime rezultata istraživanja se nalaze u šestom poglavlju. Na kraju rada se nalaze spisak korišćene literature i listinzi procedura.

U prvom poglavlju ovog rada definisani su teorijski koncepti savremenih modela baza podataka: relacionog modela, deduktivnog modela, modela entiteta i poveznika (ER modela) i objektno - orijentisanog modela podataka. U opisu deduktivnog modela podataka identifikovani su osnovni koncepti sve tri komponente ovog modela: strukturalne, integritetne i operativne komponente. Istaknute su osnovne karakteristike savremenih sistema baza podataka - aktivnih sistema. Ukratko su opisane osnovne karakteristike najpoznatijih praktično realizovanih softvera za rukovanje bazama podataka zasnovanih na relacionom modelu podataka (ORACLE) i post-relacionih aktivnih sistema (POSTGRES-a i STARBURST-a).

U drugom poglavlju su definisani osnovni didaktički koncepti u projektovanju obrazovnog računarskog softvera. Istaknute su mogućnosti primene baza podataka u projektovanju različitih tipova obrazovnog softvera u odnosu na dve karakteristične klasifikacije. Dat je prikaz modela podataka u realizaciji pojedinih elemenata, koji mogu biti prisutni u raznim tipovima obrazovnog softvera.

U trećem poglavlju su definisane teorijske osnove za realizaciju deduktivnih modela baza podataka. Opisana je metoda rezolucije, OL-rezolucija sa markiranim literalima i date su osnovne karakteristike ADT sistema. Opisane su osnovne karakteristike PROLOG-a, pri čemu je posebno istaknut problem negacije i konačnog neuspeha u PROLOG-u. U ovom poglavlju je, takođe, opisan DATALOG kao pokušaj spajanja jezika logičkog programiranja sa bazama podataka. Navedeno je da DATALOG koristi stil PROLOG-a za izražavanje pravila u formi Hornovih klauzula. Opisan je CWA-princip (princip zatvorenog sveta) na kome se temelji rad DATALOG-a.

Na osnovu ovih teorijskih koncepata definisana je osnovna hipoteza ovog istraživanja i ona se sastoji u povezivanju koncepata otvorenog i zatvorenog sveta u jedinstven sistem za rukovanje bazama podataka. Osnovna hipoteza definisana je u četvrtom poglavlju. Takođe, u ovom poglavlju definisani su pojmovi: CWA-predikata, CWA-pravila, objašnjen je princip ugrađivanja CWA-kontrolera u rezolucijske procedure i posebno, u OL-rezoluciju sa markiranim literalima. Na osnovu ovih koncepata razvijen je konkretan programski sistem BASELOG, koji je takođe objašnjen u ovom poglavlju. Poseban osvrt je dat na korisnički aspekt sistema - zadavanje upita i odgovori BASELOG-sistema. Izvršena je komparacija rada BASELOG-sistema sa ADT sistemom, PROLOG-om i DATALOG-om na osnovu konkretnih primera. Na kraju ovog poglavlja nalaze se osnovne informacije o programskoj implementaciji BASELOG-sistema i navedene su mogućnosti njegovog daljeg proširenja.

U petom poglavlju definisan je opšti koncept projektovanja obrazovnog računarskog softvera, koji koristi deduktivne baze podataka zasnovane na BASELOG-sistemu. Opisan je postupak povezivanja BASELOG-a sa konkretnom bazom podataka. Dati su primeri rada BASELOG-a na realnoj bazi znanja. Prvo su navedeni jednostavniji primeri, a zatim je naveden kompleksniji primer, sa bazom podataka u kojoj se nalazi više različitih predikata. Na tom primeru je realizovan rad BASELOG-sistema i u otvorenom i u zatvorenom režimu rada u zavisnosti od semantičke kompletnosti baze u odnosu na dati predikat. U svim ovim primerima izvršena je komparacija rada BASELOG-a sa DATALOG-om i PROLOG-om, pri čemu su istaknute prednosti koje ima BASELOG u odnosu na ova dva sistema u oblasti projektovanja i primene baza podataka u obrazovnom softveru.

U šestom poglavlju se nalaze zaključak i rezime rezultata istraživanja, kao i pravci daljeg razvoja sistema.

Zahvaljujem se svojoj porodici na podršci koju mi je uvek pružala i pruža. Izuzetnu zahvalnost izražavam profesorima Tehničkog fakulteta “Mihajlo Pupin” u Zrenjaninu: mentoru, profesoru dr Đorđu Nadrljanskom, i profesoru dr Petru Hotomskom, za ideje, motivaciju, podršku i pomoć. Takođe se zahvaljujem docentu dr Ivanu Lukoviću na korisnim i kreativnim sugestijama prilikom završne obrade disertacije. Za sugestije i pomoć pri programskoj implementaciji zahvaljujem se docentu dr Ivani Berković i Nedeljku Divjaku, kao i svima koji su mi na bili koji način pomogli u radu.

1. UVOD

U postupku projektovanja baza podataka veoma je važno poznavati koncepte odabranog modela podataka. Zbog toga se na početku ovog poglavlja definišu osnovni koncepti četiri najpoznatija modela podataka [Ullm88]: relacioni, model entiteta i poveznika (ER model podataka), deduktivni i objektno - orijentisani model. Opisane su osnovne karakteristike aktivnih sistema baza podataka i praktično realizovanih najpoznatijih softvera za rukovanje bazama podataka: relacionog (ORACLE), softvera za personalne računare, kao i prototipova aktivnih sistema (POSTGRES-a i STARBURST-a).

1.1. Pregled teorijskih koncepata modela baza podataka

Model podataka je formalno-apstraktni koncept putem koga se predstavlja semantika realnog sveta u bazi podataka. Model podataka treba da predstavi strukturu (statičke osobine), ponašanje (dinamiku) sistema i ograničenja koja važe u sistemu. Statičke osobine su relativno nepromenljive u dužem vremenskom intervalu. Ograničenjima se predstavljaju pravila ponašanja u realnom sistemu, dozvoljene i nedozvoljene vrednosti podataka i dozvoljeni i nedozvoljeni odnosi između komponenata realnog sistema. Cilj ograničenja je da se obezbedi integritet baze podataka a to znači da podaci o realnom sistemu budu usaglašeni sa ograničenjima sistema. Dinamičkom komponentom modela podataka odslikavaju se promene u realnom sistemu. Potrebno je definisati operacije koje će sadržaj baze podataka održavati u skladu sa izmenama u realnom sistemu. Definicija modela podataka se navodi prema ([ML96], str. 2):

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

U toku poslednjih trideset godina razvijeno je više modela podataka. Karakteristični modeli su: mrežni, hijerarhijski, relacioni, model entiteta i poveznika, funkcionalni model, model semantičkih hijerarhija, semantički model, objektno-orijentisani model, logički model podataka.

Hijerarhijski i mrežni model podataka bili su zastupljeni šezdesetih i sedamdesetih godina. Relacioni model podataka je predstavljen 1970. godine i od tog perioda počinje da bude najčešće korišćeni model podataka u praksi projektovanja i razvoja informacionih sistema i baza podataka zbog niza svojih dobrih osobina. Međutim, zbog nedovoljno razvijenih koncepata za predstavljanje semantike realnog sveta u relacionom modelu, savremeni pristupi u projektovanju i razvoju baza podataka idu ka razvoju modela podataka sa semantički razvijenijim konceptima: objektno-orijentisani model, semantički model, model entiteta i poveznika, logički (deduktivni) model.

U ovom radu će biti dat prikaz relacionog, deduktivnog modela, modela entiteta i poveznika i objektno-orijentisanog modela. Ovi modeli su, ujedno, najviše zastupljeni u praksi projektovanja baza podataka ([ML96], [Ma96], [PBR90], [Sto90]). Prikaz svakog modela podataka se sastoji od opisa sve tri komponente modela podataka: strukturalne (strukturne), integritetne, operacijske (operativne).

1.1.1. Relacioni model

Teorijske postavke relacionog modela podataka dao je Codd 1970. godine. Osnovu ovog modela podataka čine relacije i njihovo predstavljanje putem dvodimenzionalnih tabela. Razvoju teorije relacionog modela značajan doprinos dali su i: Beeri, Bernstein, Fagin, Maier, Rissanen, Ullman i drugi. Podsticaj za razvoj ovog modela bili su nedostaci do tada najčešće korišćenih modela: hijerarhijskog i mrežnog. Ti nedostaci se ogledaju u sledećem [ML96]:

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

Logička nezavisnost je postignuta postavljanjem jasne granice između načina prezentacije podataka u programima i načina putem koga se ti podaci fizički memorišu na medijumu eskterne memorije.

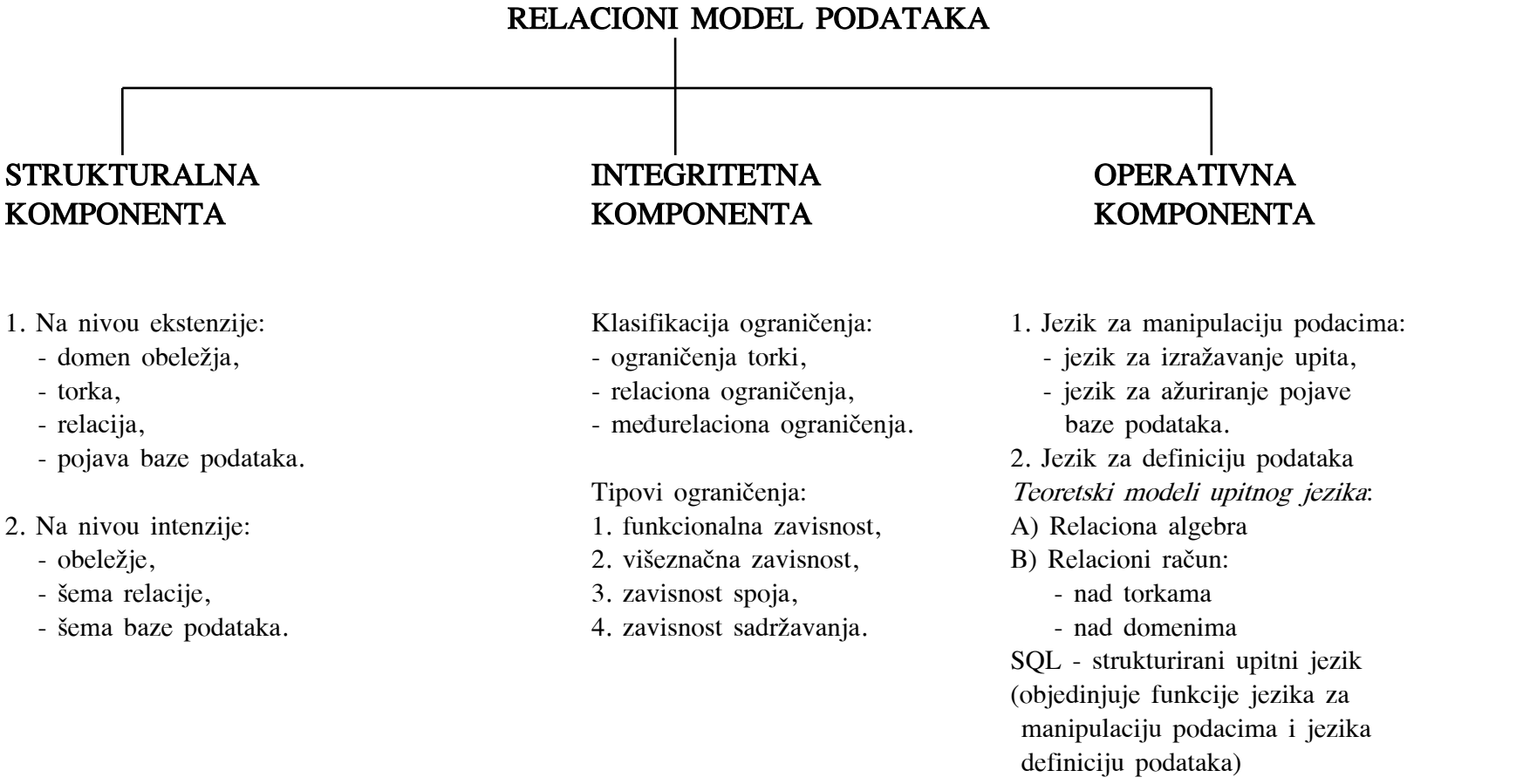
Strukturalna kompleksnost je izbegnuta uvođenjem koncepta dvodimenzionalne tabele kao reprezenta relacije. Navigacioni jezik za manipulisanje podacima relacionog modela je jezik koji omogućuje visoku nezavisnost između programa, sa jedne strane, i mašinske reprezentacije i organizacije podataka, sa druge strane.

Strukturalna komponenta relacionog modela podataka

U daljem tekstu se prema [ML96] navode osnovne definicije koncepata relacionog modela.

U opisu strukturalne komponente modela podataka koriste se dva pojma. To su: intenzija i ekstenzija. ***Intenzija*** je definicioni opis nekog skupa. Ona definiše skup navođenjem uslova koje njegovi elementi treba da zadovolje. ***Ekstenzija*** je jedna od moguće *pojave* skupa nabrajanjem elemenata. Intenzija je generalizacija skupa intenzija. Intenzija definiše zajedničke osobine svojih ekstenzija.

Na nivou ekstenzije koncepte relacionog modela podataka čine: domen obeležja, torka, relacija i pojava baze podataka. Na nivou intenzije, koncepte predstavljaju: obeležje, šema relacije i šema baze podataka.
Na slici 1.1. nalaze se predstavljeni u obliku tabele osnovni koncepti relacionog modela podataka.


RELACIONI MODEL PODATAKA
STRUKTURALNA KOMPONENTA
1. Na nivou ekstenzije:
- domen obeležja,
- torka,
- relacija,
- pojava baze podataka.
2. Na nivou intenzije:
- obeležje,
- šema relacije,
- šema baze podataka.
INTEGRITETNA KOMPONENTA
Klasifikacija ograničenja:
- ograničenja torki,
- relaciona ograničenja,
- međurelaciona ograničenja.
Tipovi ograničenja:
1. funkcionalna zavisnost,
2. višeznačna zavisnost,
3. zavisnost spoja,
4. zavisnost sadržavanja.
OPERATIVNA KOMPONENTA
1. Jezik za manipulaciju podacima:
- jezik za izražavanje upita,
- jezik za ažuriranje pojave baze podataka.
2. Jezik za definiciju podataka
Teoretski modeli upitnog jezika:
A) Relaciona algebra
B) Relacioni račun:
- nad torkama
- nad domenima
SQL - strukturirani upitni jezik
(objedinjuje funkcije jezika za manipulaciju podacima i jezika definiciju podataka)

Slika 1.1.

Definicija 1.2. Obeležja (***atributi***) su osobine entiteta (predstave realnih objekata).

Definicija 1.3. Domen obeležja je skup mogućih vrednosti koje obeležje može imati u konkretnim slučajevima.

Skup $U = \{ A_i \mid i = 1,...,m\}$ je podskup skupa obeležja realnog sistema. Skup U je konačan i najčešće se zove ***univerzalni*** skup obeležja.

Dat je skup obeležja $R \subset U$.

Definicija 1.4. Jedna ***R - vrednost***, u oznaci $t[R]$ ili kratko t ako je R poznato, je funkcija koja preslikava svako obeležje iz R u odgovarajuću vrednost, odnosno

$$t : R \to Dom,$$

gde je $Dom = \bigcup_{i=1}^{k} dom(A_i)$ i važi $t(A_i) \in dom(A_i)$ za $i = 1,...,k$.. [ML96]

Neka je $X \subseteq R$, a $t[R]$ jedna R-vrednost. ***Restrikcijom R-vrednosti*** $t[R]$ na skup obeležja X, naziva se takva X - vrednost $u[X]$, u kojoj je svakom obeležju A iz X pridružena ista vrednost kao i u R-vrednosti $t[R]$.

Kada se putem indeksa izvrši numerisanje obeležja tada se obeležje A_i opisuje kao i-to obeležje (po redu) a k-torka ili kratko torka, $(a_1,...,a_k)$ se koristi za skraćeno označavanje R - vrednosti. U tom smislu se R - vrednosti nazivaju ***torkama.***

Definicija 1.5. Relacija je svaki konačan skup R - vrednosti. [ML96]

Definicija 1.6. Šema relacije je imenovana dvojka, u oznaci $N(R,C)$, gde je N naziv šeme relacije, $R \subseteq U$ skup obeležja, a C skup ograničenja. [ML96], [PBG89]

Skup ograničenja C opisuje odnose između elemenata domena iz obeležja R.

Primer 1.1. Dat je tip entiteta UČENIK sa sledećim skupom obeležja: {MBR, IME, PREZIME, ADRESA, GOD_ROD}, pri čemu obeležje MBR ima značenje: matični broj učenika. Za ovaj entitet važi ograničenje:

(γ) svaki učenik ima jedinstveni matični broj i ne postoje dva učenika sa istim jedinstvenim matičnim brojem.

Šema relacije koja reprezentuje ovaj entitet ima sledeći oblik:

Učenik({MBR,IME,PREZIME, ADRESA,GOD_ROD}, {γ}) (1)

ili:

Učenik({MBR,IME,PREZIME, ADRESA,GOD_ROD}, {MBR}) (2)

U ovom radu će se za reprezentaciju šeme relacije koristiti oblik (2).

Definicija 1.7. Relacija r(R), koja zadovoljava svako ograničenje iz skupa C, predstavlja ***pojavu*** nad šemom relacije (R,C). Skup svih pojava nad šemom relacije (R,C) obeležava se sa $SAT(R,C)$. [ML96], [PBG89]

Definicija 1.8. Skup obeležja $X \subseteq R$ predstavlja ključ šeme relacije (R,C), ako za svako $r \in SAT(R,C)$ važe sledeća dva uslova:

1^0 $(\forall u,v \in r)\ (u[X] = v[X] \Rightarrow u = v)$ i

2^0 $(\forall Y \subset X)\ (\neg 1^0)$. [ML96]

Definicija 1.9. Šemu relacione baze podataka predstavlja imenovana dvojka $N(S,I)$, gde je N naziv šeme baze podataka, S konačan skup šema relacija, u oznaci $S=\{(R_i,C_i) \mid i=1,...,n\}$, takvih da je $U = \bigcup_{i=1}^{n} R_i$, a I predstavlja skup međurelacionih ograničenja i ograničenja, koja su posledica pravila poslovanja. [ML96]

Definicija 1.10. Pojava nad šemom baze podataka $N(S,I)$ je takva pojava s nad S, koja zadovoljava svako $i \in I$, u oznaci $s \in SAT(S,I)$. [ML96]

Integritetna komponenta relacionog modela podataka

Integritetnu komponentu predstavljaju ograničenja i ona se klasifikuju kao:

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

Najvažniji tipovi ograničenja relacionog modela podataka su:

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

S obzirom na predmet i cilj ovog istraživanja biće definisan pojam funkcionalne zavisnosti.

Definicija 1.11. Izraz oblika $X \rightarrow Y$, gde su $X \subseteq U$ i $Y \subseteq U$ proizvoljni skupovi obeležja se naziva ***funkcionalna zavisnost*** (***fd***). [Lu96], [ML96]

Definicija 1.12. Neka je r relacija nad skupom obeležja U, a X i Y dva podskupa skupa U. Skup torki relacije r ***zadovoljava funkcionalnu zavisnost*** $X \rightarrow Y$, ako za svake dve torke u i v u r važi implikacija

$$u[X] = v[X] \Rightarrow u[Y] = v[Y],$$

gde je $t[Z]$ vrednost obeležja Z u torki t. [ML96], [PBG89]

Operativna komponenta relacionog modela podataka

Operativnu komponentu čine tri podkomponente [ML96]:

- ***upitni jezik***
- ***jezik za ažuriranje podataka***
- ***jezik za definiciju podataka***.

Upitni jezik i jezik za ažuriranje podataka zovu se jednim imenom ***jezik za manipulaciju podacima*** (Data Manipulation Language).

Jezik za definiciju podataka služi za opis implementacione šeme baze podataka.

Karakteristika relacionog modela podataka je da njegovu operativnu komponentu predstavlja ***jedinstveni*** jezik podataka, koga odlikuje ***visok nivo deklarativnosti***. To znači da korisnik (projektant informacionog sistema i baze podataka, programer, krajnji korisnik) zadaje sistemu samo ***šta*** želi da uradi a ne ***kako*** (ne zadaje se ***način*** zadovoljenja cilja).

Teoretski model upitnog jezika relacionog modela podataka predstavlja ***relaciona algebra***. To je skupovno orijentisani, visoko deklarativni upitni jezik.

Definicija 1.13. Relaciona algebra je struktura

$$\mathbb{A}_r = (U, Ds, dom, \boldsymbol{S}, \text{rbp}, \Omega, O)$$

pri čemu je U univerzalni skup obeležja, Ds skup domena, $dom : U \rightarrow Ds$ domenska funkcija, $\boldsymbol{S} = (S,I)$ šema relacione baze podataka, *rbp* relaciona baza podataka nad $\boldsymbol{S}$, $\Omega = R \cup I \cup L$ skup relacionih operatora, logičkih funkcija i logičkih operatora i $O = \{\cup, \cap, -, \delta_F, \sigma, \pi, \triangleright\triangleleft, \times, [F], \div\}$. [ML96]

Simboli $\cup$, $\cap$, $-$ označavaju redom binarne operacije ***uniju, presek*** i ***razliku***; simbol δ_F označava ***preimenovanje obeležja***; simboli σ, π, $\triangleright\triangleleft$, $\times$, $[F]$, $\div$ označavaju redom

operacije: ***selekcije, projekcije, prirodnog spoja, dekartovog proizvoda, teta spoja*** i ***deljenja.***

Relacioni račun je drugi teoretski model upitnog jezika relacionih baza podataka i on se zasniva na principima ***predikatskog računa***. On se razlikuje od relacione algebre po tome
što omogućava eksplicitno definisanje tipa i konteksta promenljive, i to u odnosu na univerzalni skup obeležja *U*. Razlikuju se:

- relacioni račun ***nad torkama*** i
- relacioni račun ***nad domenima***.

Standardni upitni jezik relacionih sistema za upravljanje bazama podataka je ***SQL*** (**S**tructured **Q**uery **L**anguage). On je nastao 1974. godine. Njega odlikuje visok stepen deklarativnosti i on objedinjuje funkcije jezika za definiciju podataka, za manipulisanje podacima i funkcije upitnog jezika.

U domenu upitnog jezika, SQL se može posmatrati kao jedna implementacija relacionog računa nad torkama. Osnovna struktura SQL naredbe je [ML96]:

SELECT <*lista_obeležja*>
FROM <*lista_tabela*>
WHERE <*uslov_selekcije*>

pri čemu:

- <*lista_obeležja*> je lista obeležja, koja će se pojaviti kao rezultat upita,
- <*lista_tabela*> je lista tabela (u SQL-u termin *tabela* se koristi kao sinonim za termin *relacija*) nad kojima se vrši upit,
- <*uslov_selekcije*> je uslov za izdvajanje (selekciju) podataka iz tabela, koje su navedene iza službene reči FROM.

U [ML96], [Da87] i [Ma90] se nalaze detaljniji opisi SQL jezika.

Pored SQL upitnog jezika, relacione baze podataka imaju još jedan upitni jezik: **QBE** (**Q**uery **B**y **E**xample) - jezik postavljanja upita na osnovu primera (uzorka, obrasca). Korisnik postavlja upit tako što unosi odgovarajuće vrednosti u tabelu - uzorak relacije. Tabela-uzorak može se i redukovati u odnosu na originalnu relaciju isključivanjem kolona koje nisu potrebne. Putem specijalnih oznaka korisnik može da definiše: ili samo prikaz podataka; ili redosled prikaza podataka u rezultatu upita; spoj tabela; izmenu sadržaja tabele; brisanje torki iz tabele.

1.1.2. Deduktivni model

Deduktivni model podataka je nastao kao nadogradnja relacionog modela podataka konceptima logičkog programiranja.

Predikatske formule predstavljaju jedan od osnovnih koncepata strukturalne komponente deduktivnog modela podataka. Putem predikatskih formula (predikata) definiše se skup legalnih relacija nad skupom obeležja U (gde je skup U univerzalni skup obeležja). Promenljive predikata, u tom slučaju, predstavljaju obeležja iz podskupova $X_1,...,X_k$ skupa U.

"Neka je $X_i = \{ A_i^1,...,A_i^n \}$ podskup skupa obeležja U, za koji važi:

1^o $1 \leq |X_i| \leq |U|$, i
2^o elementi $dom(A_i^j)$, za j=1,...,n su međusobno logički povezani u realnom sistemu, tada je $P_i(X_i)$ predikat, koji opisuje relaciju r_i između elemenata domena obeležja iz X_i:

$$r_i = \{ t[X_i] \mid P_i(X_i \mid t) \}$$

pri čemu je sa $P_i(X_i \mid t)$ označena interpretacija predikata $P_i(X_i)$ s obzirom na torku t i važi da je $P_i(X_i \mid t) \in \{ \perp, T \}$

Definicija 1.14. Skup predikata $P = \{ P_1(X_1),..., P_k(X_k) \}$, takvih da je $U = \bigcup_{i=1}^{k} X_i$ definiše relaciju

$$r = \{ t[U] \mid P_1(X_1 \mid t) \wedge ... \wedge P_k(X_k \mid t) \}.$$ " ([ML96], str. 136)

To znači da relacija r nad skupom obeležja U sadrži sve one i samo one torke, koje zadovoljavaju svaki od predikata iz skupa P.

Takođe se i referenciranje na obeležja u šemi relacije, koju predstavlja dati predikat, vrši po poziciji varijabli u predikatu. Na primer, za predikat p(X,Y) iz programa, koji prezentuje relaciju OSOBA(ime, prezime) iz baze podataka, sledi da X promenljiva predstavlja obeležje "ime" a Y obeležje "prezime" date relacije.

Na slici 1.2. navedeni su koncepti koji predstavljaju komponente deduktivnog modela podataka. Detaljnije objašnjenje koncepata na slici nalazi se u **poglavlju 3.** ***Teorijske osnove za realizaciju deduktivnih modela baza podataka.***

DEDUKTIVNI MODEL PODATAKA

STRUKTURALNA KOMPONENTA

1. Term
2. Predikatske formule (IDB-predikati)
3. Torke relacione BP (EDB-predikati)
4. Klauzule (sastavci)
5. Strategije (unapred (BOTTOM-UP), unazad (TOP-DOWN), u širinu, u dubinu)

INTEGRITETNA KOMPONENTA

1. Pravila zaključivanja
2. Mehanizam zaključivanja (evaluacija predikata, rezolucijsko pobijanje)
3. Princip otvorenog i zatvorenog sveta
- konačan neuspeh (PROLOG)
- CWA-princip (DATALOG)
- CWA-pravilo (BASELOG)

4. Sigurnost pravila i izvođenja zaključaka
- zabrana negacije u glavi klauzule (PROLOG)
- sve promenljive u pravilu su limitirane (DATALOG)
- lista CWA-predikata; dužina rezolvente i dubina (odnosno širina) pretraživanja (BASELOG)

OPERATIVNA KOMPONENTA

U širem smislu:
1. Jezik predikatskog računa
- sintaksa za izražavanje pravila, činjenica i ciljeva (klauzule u PROLOG-u)
- sintaksa IDB i EDB predikata i klauzula u DATALOG-u
- sastavci, CWA-predikati i bazne aksiome u BASELOG-u

U užem smislu:
1. Jezici za rukovanje podacima u bazi podataka (koji su deo softvera za rukovanje bazama podataka ili posebno razvijeni jezici)
2. Pravila (načini) za izražavanje cilja

Slika 1.2

1.1.3. ER model podataka

ER (Entity - Relationship) model podataka se zove još i ***model entiteta i poveznika*** [ML96] ili ***model objekti-veze*** [LNB93]. Ovaj model podataka se koristi kao alat u postupku ***logičkog projektovanja*** informacionih sistema i baza podataka, jer se odlikuje bogatijom semantikom u odnosu na relacioni model podataka. Ovaj model podataka je poslužio kao osnova za razvoj ***semantičkog modela podataka***. Semantički model podataka se koristi u veštačkoj inteligenciji i u razvoju objektno-orijentisanog modela podataka.

Strukturalna komponenta ER modela podataka

Koncepti ER modela podataka na nivou intenzije su: obeležje, domen, tip entiteta i tip poveznika.
Koncepti ***obeležje*** i ***domen*** su opisani u poglavlju 1.1.1.

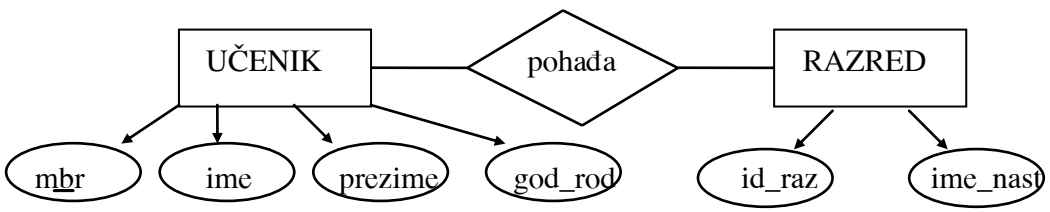
U daljem tekstu se simbolom e označava entitet, a sa $P(e)$ partitivni skup entiteta.

Definicija 1.15. Izraz oblika $N(A_1,...,A_n)$ predstavlja model skupa entiteta $E = \{e \mid P(e)\}$ i naziva se ***tipom entiteta***, ako i samo ako N predstavlja ime skupa $\{e \mid P(e)\}$, a $A_1,...,A_n$ obeležja entiteta skupa $\{e \mid P(e)\}$. [ML96]
Drugim rečima, tip entiteta predstavlja model realne klase entiteta.

Definicija 1.16. Izraz oblika $N(E_1,...,E_n;B_1,...,B_n)$ predstavlja model skupa poveznika R i naziva se ***tipom poveznika***, ako i samo ako N predstavlja naziv skupa poveznika R, $E_1,...,E_n$ su povezani skupovi entiteta, a $B_1,...,B_n$ obeležja poveznika skupa $R=\{(e_1,...,e_n) \mid e_i \in E_i,\ i = 1,...,n\}$. [ML96]

ER model podataka karakteriše grafičko predstavljanje statičke strukture realnog sistema putem takozvanih ER dijagrama. Tip entiteta se predstavlja pravougaonikom sa upisanim nazivom, tip poveznika rombom sa upisanim nazivom.

Primer 1.2. Predstaviti putem ER dijagrama sledeću realnu situaciju u informacionom sistemu škole: *učenik* (obeležja ovog tipa entiteta su: mbr, ime, prezime, god_rod) *pohađa razred* (obeležja ovog tipa entiteta su: id_raz, ime_nastavnika).



Ekstenzija ER modela podataka

Na nivou ekstenzije koncepti ER modela podataka su: pojava tipa entiteta, skup pojava tipa entiteta, pojava tipa poveznika i skup pojava tipa poveznika.
Ako se svakom obeležju tipa entiteta pridruži konkretna vrednost (podatak) iz odgovarajućeg domena dobija se ***pojava tipa entiteta***.

Definicija 1.17. Skup P je ***skup pojava tipa entiteta*** ako za svaki njegov element $(a_1,...,a_n)$ odnosno n -torku, važi da se sastoji od konkretizacije vrednosti obeležja datog tipa entiteta.

Definicija 1.18. Pojava tipa poveznika je torka koja je imenovani niz obeležja, pri čemu ta obeležja predstavljaju ključeve povezanih tipova entiteta.

Drugim rečima, dovoljno je da pojava tipa poveznika sadrži vrednosti primarnih ključeva povezanih tipova entiteta umesto kompletnih pojava povezanih tipova entiteta.

Integritetna komponenta ER modela podataka

Integritetnu komponentu ER modela podataka čine:
- integritet domena,
- kardinalnost tipa poveznika i
- slabi tip entiteta.

Integritet domena

Ovaj integritet se predstavlja putem trojke (tip podatka, dužina podatka i uslov). Tip podatka i dužina podatka ograničavaju vrednost obeležja na tip (vrstu znakova) i broj znakova (maksimalni broj znakova). Uslovi (izrazi) mogu biti regularni izrazi ili funkcije, putem kojih se takođe ograničava vrednost obeležja.

Na primer, ako se domenu *dom*(*RAZRED*) pridruži trojka (INTEGER, (1), <8) znači da obeležje *RAZRED* može uzeti vrednost iz skupa $\{0,1,..,8\}$.

Nula vrednost je specijalno ograničenje putem koga se određuje da li obeležje sme imati nedefinisanu vrednost. Značenje nula-vrednosti može biti:
- postojeća, ali nepoznata vrednost ili
- neprimereno svojstvo.

Na primer, obeležju *NAGRADE_TAKMIČENJA* tipa entiteta *UČENIK* može biti pridružena nula-vrednost, ako se radi o učeniku, koji nije učestvovao ni na jednom takmičenju.

Kardinalnost tipa poveznika

Prirodu odnosa među povezanim tipovima entiteta odslikava ***kardinalnost tipa poveznika***. Ako se posmatra binarna relacija R između skupova pojava dva tipa entiteta, onda se kardinalnost preslikavanja odnosi na brojnost (kardinalitet) elementa partitivnog skupa u koji se preslikava jedan elemenat skupa originala.

Kardinalitet relacije R, odnosno tipa poveznika R, se označava sa:

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

pri čemu su sa E_1 i E_2 označeni skupovi pojava poveznih tipova eniteta, sa a_1 i a_2 minimalni kardinalitet a sa b_1 i b_2 maksimalni kardinalitet preslikavanja.

"Parametrima a i b se najčešće dodeljuju sledeće karakteistične vrednosti:

- parametru a se dodeljuje vrednost 0, ako se bar jedan elemenat skupa originala preslikava u prazan skup, inače mu se dodeljuje vrednost *1*,
- parametru b se dodeljuje vrednost *1*, ako kardinalitet slike svakog originala nije veći od *1*, inače mu se dodeljuje vrednost N ili M, gde je $1<N,M\leq |E|$."([ML96], str. 24)

Slučaj kada je parametar a_2=1 se zove ***egzistencijalno ograničenje***, jer se tumači na sledeći način: da bi e_1 pripadao E_1 mora biti povezan najmanje sa jednim e_2 iz E_2.

Međutim, kardinalitet tipa poveznika se tumači i na sledeći način: ako postoji e_1 iz E_1 takvo da se ne javlja nijedanput kao prva komponenta para (e_1,e_2) tada je e_2=0 inače a_2=1. Ako za svako $e_1\in E_1$ važi da se javlja najviše jedanput kao prva komponenta para (e_1,e_2), tada je b_2=1, inače je b_2=N. Ako se kardinalitet tipa poveznika tumači na ovakav način, onda se kardinalitet označava sa:

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

Na primer, ako je $R_1(E_2(1,b_2))$ to znači da mora postojati neki entitet e_2 u E_2, da bi entitet e_1, koji je sa njim u vezi, mogao biti uključen u skup E_1. Ovaj tip kardinaliteta ujedno označava ***prethođenje***, odnosno situaciju u kojoj postojanje entiteta e_2 iz skupa E_2 prethodi nastanku entiteta e_1 iz E_1.

U ER dijagramima se predstavljanje kardinaliteta tipa poveznika vrši navođenjem ili para (a_1,b_1) i (a_2,b_2) ili samo maksimalnih vrednosti parametara b_1 i b_2 uz simbol, koji grafički predstavlja odgovarajući tip entiteta.

Primer 1.3. Na slici 1.3. predstavljen je ER dijagram, koji opisuje realan sistem u kojem:

- učenik pohađa u datom trenutku samo jedan razred,
- jedan razred pohađa više učenika a obavezno bar jedan učenik.

Slika 1.3.

U odnosu na maksimalne vrednosti kardinaliteta, tipovi poveznika se mogu podeliti u tri grupe: *M : N, 1 : N* i *1 : 1*.

Rekurzivni tip poveznika je tip poveznika koji predstavlja model relacije, koja povezuje entitete iste klase. Ovaj tip poveznika služi za modelovanje situacija kao što su: radnik rukovodi (drugim) radnicima, proizvod je sastavljen od (drugih) proizvoda.

Primer 1.4. Za predstavljanje rodbinskih veza otac-deca u skupu pojava tipa entiteta *OSOBA* koristi se sledeći rekurzivni tip poveznika:

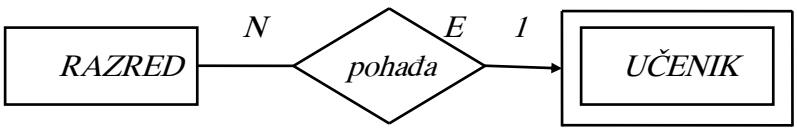
(*Osoba*(ima_oca)(0,1):*Osoba*(ima_decu)(0,N))

sa značenjem: jedna osoba ima više dece (mada ne mora imati ni jedno dete) i svaka osoba ima samo jednog oca.

U slučaju da tip poveznika povezuje više od dva tipa entiteta radi se o tipu poveznika reda većeg od dva. Postupak za određivanje kardinaliteta ovog tipa poveznika se ponavlja *n* puta, za svaki od povezanih tipova entiteta jedanput.

Ograničenje tipa $R(E_1(a_2,b_2):E_2(1,b_1))$ ukazuje na postojanje ***slabog tipa entiteta***, odnosno za svako e_2 iz E_2, postoji bar jedno e_1 iz E_1 pri čemu su e_1 i e_2 povezani relacijom R. Ovo ograničenje zove se još i ***egzistencijalnom zavisnošću*** povezanih tipova entiteta. Tip entiteta koji nije slab zove se jak (regularan) tip entiteta.

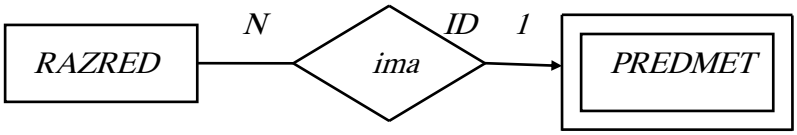
Primer 1.5. Na slici 1.4. putem slabog tipa entiteta *Učenik* i jakog tipa entiteta *Razred* predstavljena je situacija u kojoj ne može postojati učenik (u posmatranoj školi) a da ne pohađa razred.



Slika 1.4.

Tip entiteta koji ne poseduje samostalni ključ i koji je slab s obzirom na povezani tip entiteta je ***identifikaciono zavisan.***

Primer 1.6. Na slici 1.5. su nacrtana dva tipa entiteta. Tip entiteta *Predmet* je identifikaciono zavisan od tipa entiteta *Razred* (misli se na pripadnost određenih predmeta prvom, drugom, trećem..., razredu osnovne (srednje) škole u odnosu na nastavni plan).



Slika 1.5.

Operativna komponenta ER modela podataka

ER model podataka se koristi za logičko projektovanje baza podataka. Još uvek ne postoji konkretan softver za rukovanje bazama podataka, koji je zasnovan na ovom modelu. Razvijeno je nekoliko jezika ER modela podataka, koji prevode upite iskazane terminima ER modela u SQL upite. U [LuW90] je definisan ER upitni jezik i dati su primeri prevođenja upita na engleskom jeziku u ovaj upitni jezik. U ovom ER upitnom jeziku uključeni su i koncepti IS_A i IS_PART_OF hijerarhija, da bi se pokazalo na koji način semantički model podataka podržava upite na prirodnom jeziku.

U [PS85] se navodi da ne postoje jasno definisane osnovne operacije za jezik za manipulaciju podacima (DML) za ER model. Ne postoji jezik, koji u potpunosti podržava sve mogućnosti ER modela sa n-arnim tipovima poveznika, tipovima poveznika sa atributima, kompleksnim i višeznačnim atributima. U tom radu se predlaže jedan skup operatora za ER model podataka, koji je u vezi sa relacionom algebrom (DML jezikom relacionog modela podataka).

U [CER90] predlaže se grafički pristup reprezentacije logičke strukture baze podataka. Ideja rada je da se korisniku omogući da grafički formuliše svoje upite i da ih modifikuje - putem manipulacije dijagrama šeme baze podataka. Razvijen je interfejs za manipulisanje dijagramima za prošireni ER model (Extended ER model). Podržani su koncepti: generalizacije, specijalizacije (uključujući podskupove), unija i podela poveznika na dijagramima. Tako definisan model, autori ovog rada zovu *extended conceptual entity-relationship (ECER) model.* Ovaj interfejs podržava sledeće operatore ECER modela: brisanje, prikaz objekata, uniju, razliku, presek, preimenovanje, kreiranje tipa poveznika; dodavanje novog tipa entiteta na dijagramu, dodavanje tipa poveznika, brisanje tipa

entiteta i tipa poveznika i modifikaciju atributa. Ovakav pristup može biti upotrebljen kao krajnji korisnički grafički interfejs ka relacionom softveru za rukovanje bazama podataka.

U [MMT89] je opisan integrisani jezik opšte namene (ERLAN). Ovaj jezik omogućuje direktan rad sa konceptima ER modela sa mogućnošću nadogradnje nad standardne programske jezike: Cobol, Pascal, C. Namera je da se omogući pristup ER podacima iz aplikacija pisanih na pomenutim programskim jezicima. Tako integrisani jezik mora imati sledeće osobine:

- mogućnost da se radi sa strukturama podataka u bazi podataka i u aplikaciji na konzistentan način (atributi entiteta u bazi podataka moraju odgovarati varijablama u programskom jeziku, odnosno oni mogu biti korišćeni u izrazima selekcije (upita), pozivima procedura, na isti način kao i promenljive iz programa),
- sistem treba da upravlja kompleksnim višestrukim obraćanjima jednom istom entitetu
- pristup bazi podataka mora biti transparentan za programera. Programer treba da radi sa strukturama podataka iz baze, na isti način kao da su one u operativnoj memoriji.

Osim ERLAN-a, razvijeno je još nekoliko upitnih jezika za ER bazu podataka: GORDAS, ERROL, DESPATH, NETUL, EAS-e. Međutim, oni se, uglavnom, posmatraju kao krajnji interfejs nekog postojećeg relacionog softvera za rukovanje bazama podataka.

Koncepti proširenja ER modela podataka

ER model podataka se odlikuje semantički bogatijim konceptima u odnosu na relacioni model. Koncepti kojima se postiže semantičko bogatstvo ovog modela su:

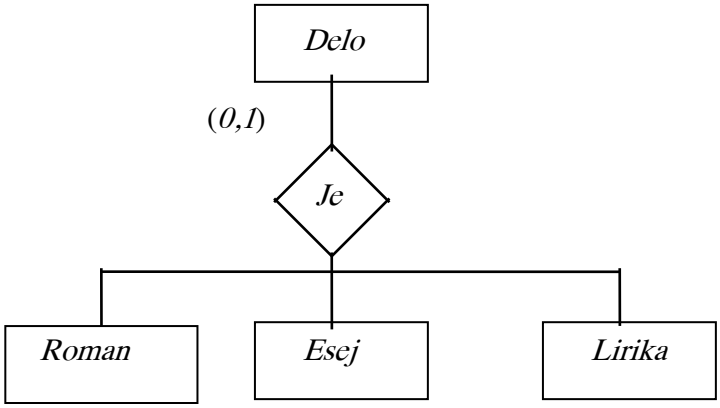
- potklasa i superklasa odnosno ***IS_A*** **hijerarhija,**
- kategorija i kategorizacija,
- gerund.

IS_A hijerarhija

Putem *IS_A* hijerarhije se identifikuju pravi podskupovi entiteta sa specifičnim osobinama ili ulogama na taj način što se zajednička obeležja svih entiteta grupišu u ***superklasu*** a specifična obeležja se grupišu u odgovarajuće ***potklase***.

Pojava potklase sadrži samo vrednost primarnog ključa superklase i vrednosti specifičnih obeležja. Svakoj pojavi potklase odgovara tačno jedna pojava superklase. Potklasa nasleđuje osobine svoje superklase.

Primer 1.7. Na slici 1.6. putem *IS_A* hijerarhije predstavljena je superklasa *Delo* sa svojim potklasama: *Roman, Esej, Lirika.*



Slika 1.6.

Na slici je označen samo kardinalitet preslikavanja sa skupa pojava superklase na skup pojava potklase. Minimalni kardinalitet ovog preslikavanja je 0, zbog toga što postoje pojave superklase kojoj ne odgovoraju ni jedna pojava potklase (među delima može biti i epsko delo, naučna fantastika i dr.). Maksimalni kardinalitet preslikavanja je *1*, zbog toga što svakoj pojavi superklase odgovara pojava iz najviše jedne potklase (delo je ili roman ili esej ili lirika).

Kardinalitet preslikavanja sa skupa pojava potklasa na skup pojava superklase se ne navodi, zbog toga što svakoj pojavi potklase odgovara uvek jedna i samo jedna pojava superklase.

Postupci, putem kojih se definiše *IS_A* hijerarhija su: specijalizacija i generalizacija. Postupak specijalizacije počinje od tipa entiteta buduće superklase, pa se na osnovu klasifikacionog obeležja, izdvajaju potklase sa specifičnim obeležjima. Postupak generalizacije je suprotan: polazi se od različitih tipova entiteta, pa se identifikacijom zajedničkih osobina formira zajednička superklasa.
Cilj uvođenja *IS_A* hijerarhije je identifikacija različitih uloga entiteta u realnom sistemu i prirodnije predstavljanje veza između entiteta. Takođe, na nivou ekstenzije izbegavaju se nula-vrednosti za specifična obeležja, u slučaju specijalizacije, ili eliminisanje ponavljanja istih podataka, u slučaju generalizacije.

Kategorizacija je postupak izgradnje modela u kome potklasa objedinjuje pojave potpuno različitih tipova entiteta i tada se potklasa zove ***kategorijom***.

Gerund je koncept putem koga se predstavlja situacija u kojoj su (ne nužno) sve pojave tipa poveznika povezane sa pojavama nekog drugog tipa poveznika. U tom slučaju se tip poveznika transformiše u gerund i zato se grafički predstavlja kombinacijom simbola za

predstavljanje tipa entiteta i tipa poveznika (pravougaonik u kome je simbol romba). U [LNB93] se ovaj koncept zove ***mešoviti tip entitet-veza***.

Na slici 1.7. predstavljeni su osnovni koncepti ER modela podataka.

1.1.4. Objektno-orijentisani model

Primene baza podataka u oblastima: računarom podržano projektovanje, multimedijalni sistemi, baze znanja imale su za posledicu nastanak i razvoj objektno-orijentisanog modela podataka. Relacioni model podataka nije mogao da zadovolji zahteve ovih oblasti primene kao što su: rad sa nehomogenim skupovima podataka, intenzivne izmene šeme baze podataka, upravljanje različitim verzijama jednog objekta, upravljanje dugačkim transakcijama. [ML96]

Objektno-orijentisani model podataka nije precizno definisan. On se posmatra ili kao nadogradnja nad relacioni model podataka ili kao nadogradnja nad objektno-orijentisane programske jezike [PBR90]. U [ML96] se navodi da je objektno-orijentisani model podataka nastao integracijom karakteristika objektno-orijentisanih programskih jezika, semantičkog modela i relacionog modela podataka. Zbog toga se često koristi termin ***paradigma*** (umesto modela podataka).

Definicije osnovnih koncepata: objekat, metoda, poruka, inkapsulacija, klasa, nasleđivanje i identitet objekta su preuzete iz [ML96].

Definicija 1.19. Neka je U skup obeležja, a $D = \{$ceo broj, realan broj, niz karaktera fiksne ili promenljive dužine, datum, novac,...$\}$ skup osnovnih tipova objekata. Tada važi:

1^0 Svaki elemenat svakog osnovnog tipa podataka je ***primitivni objekat***.
2^0 Ako su: $X_1,...,X_n$ različita obeležja iz U, a $o_1,...,o_n$ objekti, tada je

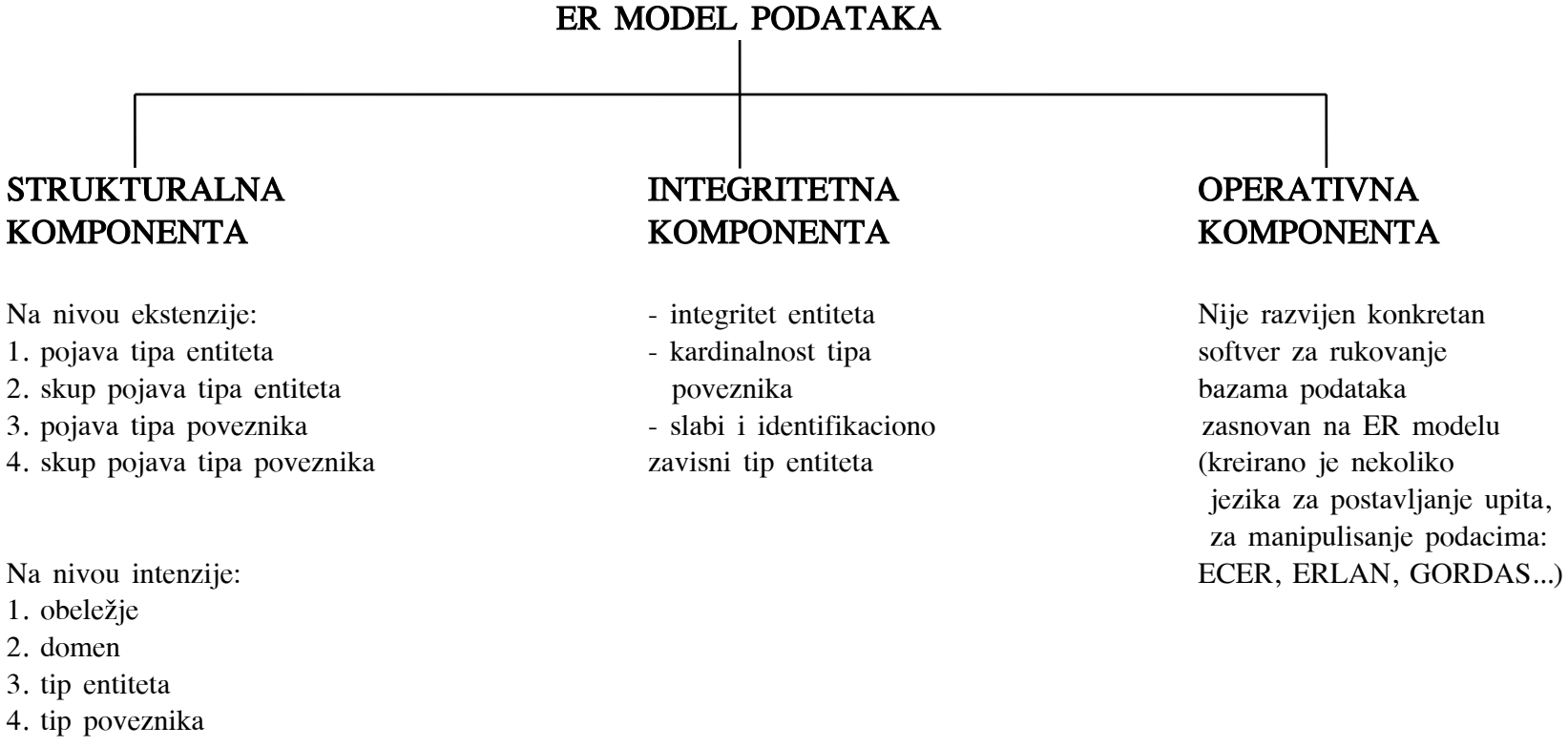
$$o = ((X_1,o_1),...,(X_n,o_n))$$

jedan ***objekat-torka***. Objekat o_i je vrednost obeležja X_i, u oznaci $o_i = X_i$.
3^0 Ako su $o_1,...,o_n$ različiti objekti tada je

$$o = \{o_1,...,o_n\}$$

jedan ***objekat-skup*** i važi $o_i \in o$. ([ML96], str. 226)

Stanje objekta se reprezentuje putem strukture nad skupom podataka a ponašanje objekta je skup procedura, koje se nazivaju ***metodama*** ili ***operacijama***.
Poruka je zahtev, upućen objektu, da izvrši metodu.


ER MODEL PODATAKA
STRUKTURALNA KOMPONENTA
Na nivou ekstenzije:
1. pojava tipa entiteta
2. skup pojava tipa entiteta
3. pojava tipa poveznika
4. skup pojava tipa poveznika
Na nivou intenzije:
1. obeležje
2. domen
3. tip entiteta
4. tip poveznika
INTEGRITETNA KOMPONENTA
- integritet entiteta
- kardinalnost tipa poveznika
- slabi i identifikaciono zavisni tip entiteta
OPERATIVNA KOMPONENTA
Nije razvijen konkretan softver za rukovanje bazama podataka zasnovan na ER modelu (kreirano je nekoliko jezika za postavljanje upita, za manipulisanje podacima: ECER, ERLAN, GORDAS...)

Slika 1.7.

Inkapsulacija je princip skrivanja informacija, po kome svaki objekat ima svoj ***privatni*** i ***javni*** deo. Privatni deo objekta čine struktura podataka i skup metoda za tu strukturu podataka i ovaj deo objekta se zove ***implementacijom***. Javni deo objekta čine identifikator i nazivi metoda i ovaj deo objekta zove se ***interfejsom***.

Definicija 1.20. Neka je U skup obeležja, a $D = \{$ceo broj, realan broj, niz karaktera fiksne ili promenljive dužine, datum, novac,...$\}$ skup osnovnih tipova objekata. Tada važi:

1^0 Dvojka (A, d), gde je $d \in D$, je ***tip objekta***.
2^0 Ako su: $T_1,...,T_k$ tipovi objekata, tada je i $((X_1,T_1),...,(X_k,T_k))$ ***tip objekta torka.***
3^0 Ako je T tip objekta, tad je i $\{T\}$ ***tip objekta skup***. Objekat tipa $\{T\}$ je skup objekata tipa T. ([ML96], str. 230)

Klasa je dvojka (***tip objekta, skup metoda***).

Nasleđivanje je koncept objektno-orijentisane paradigme putem koga je realizovana ***hijerarhija klasa*** odnosno "je podvrsta" odnos među klasama. Klasa, koja nasleđuje osobine od druge klase, zove se ***potklasa***. Klasa, čije osobine nasleđuje jedna ili više drugih klasa, zove se ***superklasa***. Potklasa nasleđuje osobine (tip strukture i metode) od svoje superklase.

Potklasa može naslediti od svojih superklasa obeležja, metode i interfejs. Nasleđivanje metoda vodi deljenju odnosno ponovnoj upotrebi programskog koda. ***Preklapanje naziva metoda*** omogućava da se metode sa istim imenom, ali različitim značenjem i implementacijom primene na objekte različitog tipa. ***Polimorfizam*** znači da se iste (ne samo po imenu) operacije mogu primeniti na objekte različitog tipa. To znači da preklapanje naziva metoda predstavlja jedan oblik polimorfizma.

Identitet objekta

Identitet objekta se realizuje putem jedinstvenog ***identifikatora***. U objektno-orijentisanim jezicima to je pokazivač - memorijska adresa lokacije, u kojoj se nalazi objekat. Između pojma identiteta i pokazivača postoji suštinska razlika, jer je identitet objekta semantički koncept. Postupci za realizaciju identiteta su [ML96]:

- virtuelne i fizičke adrese,
- korisnički nazivi i
- surogati.

Integritetna komponenta objektno-orijentisanog modela podataka

Nula vrednost

Za objektno-orijentisani model podataka je karakteristično da se nula vrednost ili navodi eksplicitno u deklaraciji strukture tipa objekta klase ili se rešava u okviru metoda za konstruisanje objekata klase.

Integritet entiteta

Za razliku od relacionog modela podataka, gde primarni ključ ima ulogu održanja integriteta entiteta i pod kontrolom je korisnika, u objektno-orijentisanom modelu identifikator nije pod kontrolom korisnika i ne nosi nikakvu semantiku o objektu. Identifikator objekta nije adekvatan pojmu ključa u relacionom modelu. Prilikom definisanja domena obeležja potrebno je navesti vrednost "unique" ili "required" za vrednosti identifikatora objekta. Moguće je kontrolu integriteta ostvariti i putem generisanja odgovarajućeg objekta, koji sadrži informaciju o ekstenziji posmatrane klase.

Referencijalni integritet

Referencijalni integritet je posledica same strukture objekta i nema potrebe za posebnim mehanizmom za njegovo održanje. Međutim, kontrola referencijalnog integriteta između objekata potklase i superklase predstavlja još uvek nerešeno pitanje. Zbog toga što ne postoji nasleđivanje objekata, nije podržano ni nasleđivanje sa skupa objekata potklase u skup objekata superklase.

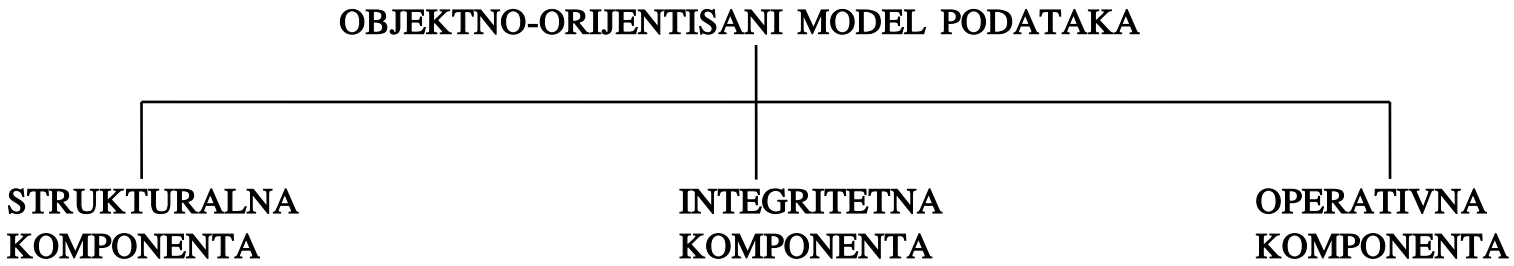
Integritet domena

Jedan od načina za kontrolu integriteta domena je definicija potklase neke postojeće klase, čiji objekti će uzimati samo dozvoljene vrednosti. Tako definisana potklasa može da se proglasi za domen obeležja.

Operativna komponenta

S obzirom da se objektno-orijentisani model podataka posmatra kao ekstenzija nad objektno-orijentisane programske jezike, to se za definisanje operativne komponente uzima sintaksa tih jezika. Međutim, ti jezici nisu u pravom smislu jezici podataka za deklarisanje struktura i za manipulisanje objektno-orijentisanim bazama podataka.

Na slici 1.8. navedeni su osnovni koncepti objektno-orijentisanog modela podataka.


OBJEKTNO-ORIJENTISANI MODEL PODATAKA
STRUKTURALNA KOMPONENTA
- objekat
- metoda
- poruka
- inkapsulacija
- klasa
- nasleđivanje
- identitet objekta
INTEGRITETNA KOMPONENTA
- nula vrednost (eksplicitno se navodi)
- integritet entiteta (nije pod kontrolom korisnika)
- referencijalni integritet
- integritet domena
OPERATIVNA KOMPONENTA
Objektno-orijentisani programski jezici (C++, Smalltalk)
To nisu jezici za deklarisanje struktura i za manipulisanje podacima
- klase
- preklapanje operatora
- nasleđivanje
- generički mehanizam

Slika 1.8.

1.2. Aktivni sistemi baza podataka

Uključivanje aktivnih komponenti sreće se u programskim jezicima (obrada izuzetaka - exceptions), a u sistemima veštačke inteligencije - demoni. Komercijalni relacioni sistemi podržavaju koncepte iskaza i trigera (ORACLE, INGRES). Savremeni istraživački projekti idu u dva pravca: izrada velikih sistema produkcionih pravila a drugi je izrada aktivnih sistema za upravljanje bazama podataka.

1. Veliki sistemi produkcionih pravila imaju mehanizam zaključivanja. Obrada činjenica i pravila se odvija u "prepoznaj-uradi" ciklusu. Prepoznavanje se sastoji u uparivanju leve strane pravila sa podacima u radnoj memoriji, a drugi deo u izvršavanju desne strane odabranog pravila. Za prepoznavanje se umesto tehnika veštačke inteligencije koriste tehnike optimizacije upita.

Za ove sisteme je karakteristično da se i podaci i pravila nalaze u bazi podataka, odnosno, omogućeno je zaključivanje nad trajnom bazom podataka, što znači da podaci i pravila ne moraju biti u radnoj memoriji. Nedostatak ovih sistema je ograničena mogućnost konkurentnog rada i oporavka. Najpoznatiji sistemi su: KBMS, RPL, RDL, ARIEL.

2. Aktivni sistemi za upravljanje bazom podataka koriste pravila kao mehanizam unifikacije za kontrolu integriteta, kontrolu prava pristupa, formiranje i izvršavanje pogleda, formiranje trigera. Izvršavanje pravila se pokreće podacima, odnosno nastankom događaja.

Pravila su sastavni deo šeme baze podataka i obrađuju se tehnikama optimizacije upita, koje su ugrađene u sistem za upravljanje bazom podataka. Konkurentan rad i oporavak su potpuno podržani kroz karakteristike softvera za rukovanje bazama podataka.

Aktivni sistemi se razvijaju ili kao nadogradnja nad objektno - orijentisane (OO) sisteme ili relacione sisteme. Najpoznatiji OO sistem, koji podržava ECA pravila je HiPAC (pored njega to su još i: O2, ODE, ETM). Najpoznatiji aktivni sistemi kao nadogradnja nad relacione sisteme su POSTGRES i STARBURST.

Pravila služe kao sredstvo unifikacije, odnosno kao mehanizam za kontrolu integriteta, definisanje materijalizovanih pogleda, kontrolu autorizacije, izradu aktivnih sistema, realizaciju deduktivnih baza podataka, sistema zasnovanih na znanju i ekspertnih sistema, održavanje verzija, memorisanih procedura.

Krajem osamdesetih godina vršena su istraživanja u pravcu spajanja baza podataka i logičkog programiranja, odnosno podataka i pravila, koja obrađuju te podatke. Rezultati tih istraživanja su sledeće tehnike:

(1) Pravila su u aplikativnim programima, a podaci u bazama podataka.

Ova tehnika je zastupljena u poslovnoj obradi podataka. Njen nedostatak je u tome što je za svako novo pravilo potrebno menjati interaktivne programe za ažuriranje baze podataka ali tako da se očuva konzistentnost baze podataka. Time se povećava složenost programa i složenost postupka održavanja baze podataka u konzistentnom stanju.

(2) Podaci i programi se nalaze u okruženju ekspertnog sistema (shell-u).

Takvo okruženje čine Prolog, KEE ili PC+. Nedostatak ove tehnike je što se zahteva da svi podaci i pravila, kojima se podaci obrađuju, budu u operativnoj memoriji, što je praktično nemoguće.

(3) Pravila se nalaze u ekspertnom okruženju a podaci u bazi podataka.

Ova tehnika je realizovana u KEE/Connection sistemu. Baza podataka je kreirana putem nekog softvera za rukovanje bazama podataka, a pravila se nalaze u okruženju ekspertnog sistema (shell-u). Ova dva sistema su "labavo" povezana na taj način što ekspertno okruženje koristi mehanizme softvera za rukovanje bazama podataka na isti način kao i svaki drugi korisnik tog softvera. Nedostatak ove tehnike je što se ona ne može primeniti u situacijama u kojima se dešavaju česte izmene podataka u bazi, jer se u tom slučaju vrši neprekidno selektovanje podataka što prouzrokuje bespotrebno trošenje resursa.

(4) Pravila i podaci se nalaze u jedinstvenoj bazi pravila i podataka.

Ova tehnika je implementirana u aktivnim sistemima baza podataka: Postgresu, Starburst-u i HiPac-u [Mar93]. Ova tehnika se zove "čvrsto povezivanje" i sastoji se u izradi jedinstvenog sistema koji će na identičan način predstavljati i obrađivati podatke i pravila.

Savremene verzije softvera za rukovanje bazama podataka (npr. ORACLE) u varijanti client/server arhitekture omogućavaju kombinaciju (1) i (4):

a) server: baza podataka, procedure i trigeri,
b) client: procedure i pravila u aplikativnim programima.

U daljem tekstu su detaljnije objašnjena dva sistema: Postgres i Starburst.

1.3. Praktično realizovani softveri za rukovanje bazama podataka

"Sistem za upravljanje bazom podataka je programski proizvod koji omogućuje efikasno: formiranje, korišćenje i menjanje baze podataka." ([ML96], str. 62). Osnovne karakteristike svakog softvera za rukovanje bazama podataka (SRBP) su:

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

U [Mi92] su navedeni osnovni zahtevi, koji se postavljaju pred sistem za rukovanje bazom podataka:

- sistem treba da održava opis implementacione šeme baze podataka (konceptualne šeme, eksterne i interne šeme) - rečnik podataka;
- lako pretraživanje podataka (SQL standard);
- obezbeđenje konkurentnog rada za više korisnika (očuvanje konzistentnosti podataka prilikom upita ili provođenja promena);
- transakcioni rad - kao metod za obezbeđenje integriteta baze podataka;
- zaštita od nenormalnih završetaka transakcija i automatsku rekonstrukciju podataka i očuvanje integriteta baze;
- upit nad distribuiranim bazama, istovremeni upit nad bazom od strane više korisnika u mreži.

U ovom poglavlju su navedene osnovne karakteristike jednog od najpoznatijeg relacionog softvera za rukovanje bazama podataka: **ORACLE**, dat je kratak prikaz okruženja za razvoj baza podataka na personalnim računarima: **FoxPro i Access**, i opisane su osnovne karakteristike dva prototipska aktivna sistema za rukovanje bazama podataka, nastala kao nadogradnja nad relacione sisteme: **POSTGRES i STARBURST**.

1.3.1. ORACLE

ORACLE RDBMS (Relational Data Base Management System) je prvi komercijalni softver za rukovanje bazama podataka. Proizvodi ga Oracle Corporation iz Belmonta u Kaliforniji. Prvobitna ideja u razvoju ovog softvera je bila razviti RDBMS koji će

koristiti strukturirani upitni jezik (SQL) i koji će imati mogućnost prenosivosti aplikacija i podataka na razne platforme.
Prva verzija ORACLE 2 se pojavila 1979. godine i radila je na DEC PDP-11 miniračunaru. Kasnije je ORACLE bio razvijen i za VAX računar pod VMS operativnim sistemom. Ova verzija imala je kompletnu implementaciju SQL-a i CONNECT BY klauzulu SELECT naredbe za rekurzivne upite kroz stablo. Verzija 3, koja se pojavila 1983. godine, bila je napisana većim delom u programskom jeziku C, što je i omogućilo implementaciju ORACLE sistema na raznim platformama. Ova verzija je imala mogućnost transakcionog rada, ali konzistentnost pri čitanju nije bila podržana. Verzija 4 pojavila se 1984. godine za IBM računare za MVS operativni sistem kao i verzija za personalne računare. Verzija 4 je omogućila konzistentnost pri čitanju i mogućnost da se prilikom upita vide podaci, koji su u saglasnosti sa vremenom kad je upit postavljen.

Verzija 5 (1985/86. godine) uvodi klijent/server arhitekturu putem mrežnog softvera SQL*Net [Ni96]. Verzija 5.1 omogućila je distribuirane upite, odnosno mogućnost pristupa podacima koji su fizički smešteni na više lokacija. Verzija 5 je prvi proizvod koji je pod MS DOS operativnim sistemom koristio *extended* memoriju personalnog računara. Verzija 6 je imala module za upravljanje baferima, konkurentni pristup, *on line backup* (rezervna verzija baze). Uvedeno je zaključavanje na nivou sloga.

Verzija 6.2 pojavila se 1991. godine i to je bio prvi višekorisnički LAN SQL server baze za operativne sisteme: OS/2, Xenix, Banyan, VINES i Macintosh. Najznačajnije poboljšanje bio je proceduralni jezik PL/SQL. To je proceduralni jezik u kome se mogu direktno pisati SQL naredbe, i ujedno je i jezik klijenta i jezik servera. Ova verzija je sintaksno podržala ANSI/ISO standard za definiciju pravila integriteta (poslovnih pravila). Pravila se zapisuju u rečnik podataka, kao i relacije između tabela.

Verzija 7 pojavila se 1992. godine. Poboljšanja u ovoj verziji su: mogućnost aplikacijama da koriste istu kopiju SQL naredbe ili procedure iz baze (*stored procedure*), koja se već nalazi u memoriji; optimizacija upita putem statistike o tabelama i indeksima; mehanizam trigera i višestruki bazni trigeri istog tipa nad jednom tabelom; metode za upravljanje zaštitom (lozinke i autorizacija); globalna optimizacija distribuiranih upita i distribuiranih ažuriranja; ažuriranje istih podataka na različitim lokacijama; mogućnost definisanja SQL funkcija (korisnički definisane funkcije); deklarativno definisanje referencijalnih integriteta.

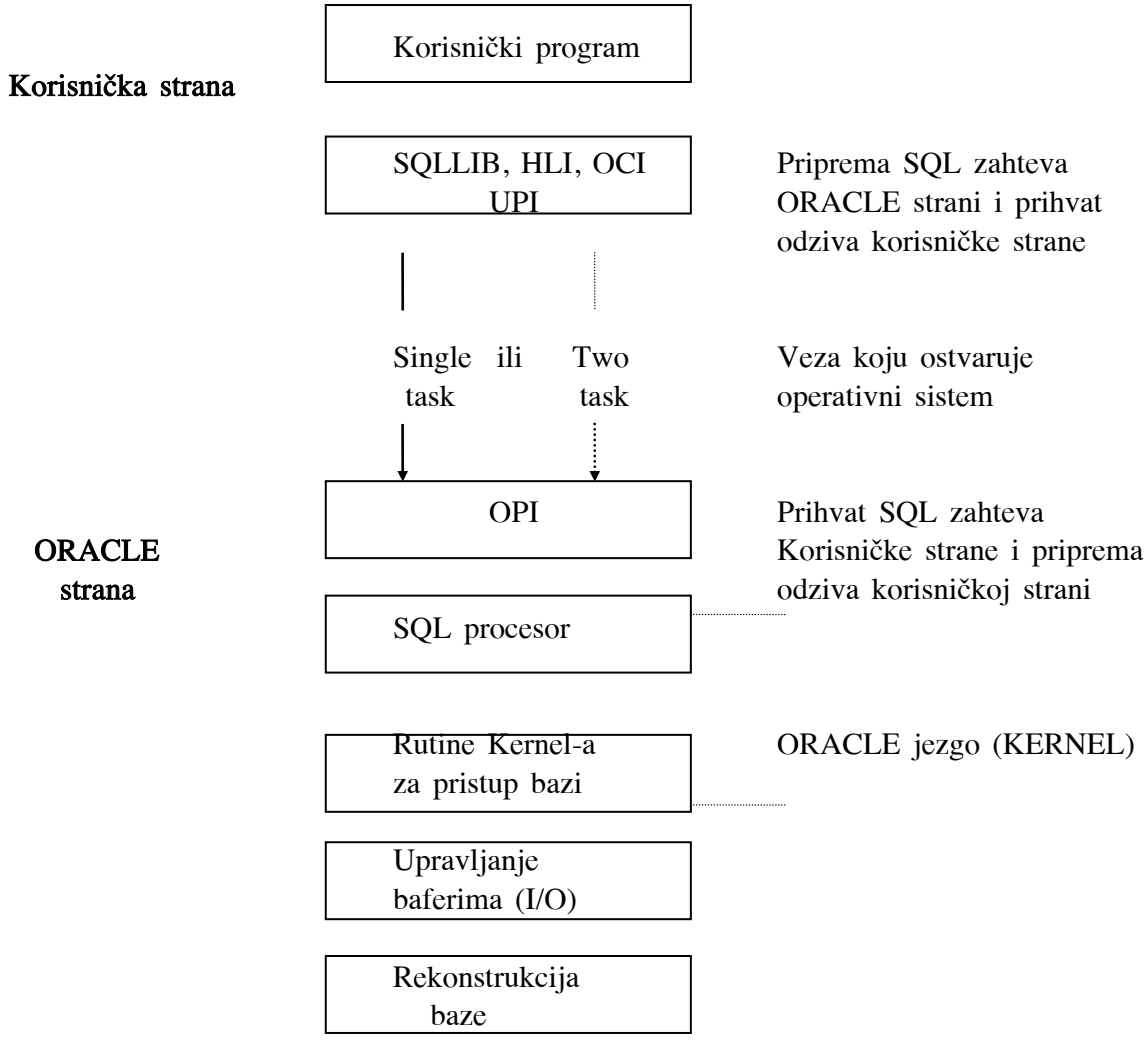
U [Rač97] navede se 12 pravila, koje prema E.F. Codd-u, tvorcu teorije relacionog modela podataka, treba da ispuni softver za rukovanje bazama podataka da bi mogao da se zove ***relacioni***. Codd je 1985. godine istakao da po tim pravilima ni jedan od postojećih softvera nije u potpunosti relacioni. U odnosu na analizu datu u [Rač97] *ORACLE 7* ne ispunjava sledeće zahteve:

- ORACLE dozvoljava postojanje tabele za koju ne postoji definisan primarni ključ, a jedno od pravila kaže da se svakom podatku može pristupiti samo kombinacijom imena tabele, vrednosti primarnog ključa i imena kolone;
- nema mogućnost ažuriranja pogleda, koji spajaju više tabela (u verziji 8 bi trebalo da se realizuje);
- posledica prethodnog nedostatka je delimična logička zavisnost programa i aplikacija od podataka.

Zadovoljenje skupa zahteva, koji se postavljaju pred svaki sistem za rukovanje bazama podataka, inicira njihovu veoma složenu programsku realizaciju. Osnovu arhitekture ORACLE sistema čine:

- korisnički deo sistema (aplikacije)
- ORACLE sistem (sistem upravljanja bazom i sama baza podataka).

Na slici 1.9. prikazana je osnovna arhitektura ORACLE sistema ([Mi92], str.5):



Slika 1.9.

U zavisnosti od operativnog sistema, korisnički proces i ORACLE proces mogu da rade kao jedinstven proces ('single task'- MS DOS, VAX/VMS, VM/CMS) ili kao posebni procesi ('two task' arhitektura - UNIX, MVS).

ORACLE sistem podržava rad u lokalnoj mreži i pristup bazama podataka iz mreže ili rad sa distribuiranom bazom podataka. Operativni sistem u tom slučaju obezbeđuje razmenu zahteva i podataka između korisničke i ORACLE strane.

ORACLE sistem se sastoji iz više programskih modula, a svaki programski modul je jedan ili više uslužnih programa. Programski moduli su:

- **ORACLE osnovni modul** se sastoji od sledećih uslužnih programa [Pol91]:

  -ODL (ORACLE Data Loader) - omogućava korisniku unošenje podataka iz standardnih datoteka,

  -EXP/IMP (Export/Import) - omogućava korisniku da premešta podatke između tabela prilikom arhiviranja podataka ili prenosa podataka između različitih operativnih sistema ili ORACLE baza podataka,

  -ODS (ORACLE Display System) - omogućava administratoru baze podataka prikaz korisničkih i ORACLE procesa,

  -AIJ (After Image Journaling) - omogućava administratoru baze podataka snimanje promena nad podacima, zbog potreba oporavka u slučaju pada sistema,

  -SGI (System Global Information) - omogućava administratoru baze podataka podelu korišćene memorije,

  -IOR Program - omogućava administratoru baze podataka da pokreće, zaustavlja i inicira sistem,

  -CCF (Create Contiguous File).

- **Easy ORACLE** - modul za vođenje korisnika u toku rada

- **SQL ORACLE moduli**:

  SQL *Plus - komandni interfejs za SQL komande
  SQL *Forms - generator formi
  SQL *Graphics - modul za grafičko prikazivanje informacija
  SQL *Net - modul za pristup bazama podataka na drugim računarima
  SQL *Report - modul za oblikovanje izveštaja

- Standardni moduli za pristup bazi podataka iz programskog jezika visokog nivoa (Pro* C, Pro* COBOL, Pro* Pascal, Pro* FORTRAN).

SQL (Structured Query Language) je standardni relacioni upitni jezik koji je od 1986. godine po ANSI standardu standard upitnih jezika. Nastao je u IBM istraživačkoj laboratoriji u San Hose-u, Kalifornija 1974. godine.

SQL je implementiran u više od 50 komercijalno raspoloživih sistema za upravljanje bazama podataka relacionog tipa (DB2, SQL/DS, ORACLE, INGRES, PROGRESS,...). Osnovne karakteristike SQL-a su:

- jednostavnost i jednoobraznost u toku korišćenja

Tabela se kreira jednom naredbom i posle kreiranja tabela je odmah raspoloživa za korišćenje. Svi podaci su memorisani u obliku tabela i rezultat bilo koje operacije se logički prikazuje u vidu tabele.

- mogućnost interaktivnog i klasičnog (aplikativnog) programiranja

Upotrebom SQL-a dobijaju se odgovori na trenutne zahteve ili je moguće SQL blokove ugraditi u aplikacije, razvijene na nekom od viših programskih jezika (C, Cobol, FORTRAN)

- neproceduralnost (proceduralnost u minimalnom stepenu)

SQL je u velikoj meri neproceduralan, jer definiše ŠTA a ne KAKO; koji podaci se žele, koje tabele se referenciraju i koji uslovi treba da budu ispunjeni, bez specifikacije procedura za dobijanje željenih podataka.

Jezgro (KERNEL) ORACLE-a izvodi SQL operacije koje zahtevaju korisnici. Ovaj deo izvodi čitanje i pisanje u datoteke baze podataka (DB datoteke - sadrže korisničke tabele i rečnik podataka) i u datoteke pre slike (BI datoteke - slike podataka pre promene). Jezgro ORACLE-a, takođe, održava podatke u SGA području (opštem sistemskom području, koje je centar aktivnosti ORACLE sistema), i koordinira aktivnost više korisnika.

Rečnik podataka je skup relacija i pregleda, kreiranih od strane ORACLE sistema. Sadrži administrativne podatke o korisnicima, relacijama i drugim objektima, i nalazi se na centralnom mestu u operacijama ORACLE-a.

AKCIJE - TRIGERI (OKIDAČI)

Putem trigera uključena je aktivna komponenta u ORACLE relacioni sistem. Triger se sastoji od definicije događaja, koji izazivaju izvršenje akcija, ako su se stekli dati uslovi. To su imenovani PL/SQL blokovi naredbi, koji se nalaze u rečniku podataka, i koji se implicitno izvršavaju kada se desi neki događaj u sistemu. Događaji, koji izazivaju izvršavanje akcije trigera odnose se isključivo na bazu podataka; uslovi, pod kojima će akcija biti izvršena proveravaju stanje baze podataka, i na kraju, akcije, koje se izvršavaju su iskazane u terminima operacija nad bazom podataka. Znači, svi elementi trigera odnose se isključivo na elemente baze podataka.

Svaki definisani triger se pridružuje jednoj tabeli, pri čemu se navodi koje od operacija dodavanja, brisanja ili ažuriranja pokreću (okidaju) izvršavanje takvog programa.

Triger je objekat baze podataka, koji se izvršava neposredno pre (BEFORE) ili posle (AFTER) operacija ažuriranja (INSERT, DELETE, UPDATE) ali pre naredbe COMMIT, koje su specificirane u samom trigeru. Triger mora da sadrži navedenu bar jednu operaciju ažuriranja. Ako se ne navede FOR EACH ROW opcija, triger se izvršava jednom za pokrenutu akciju (naredbu), a inače za svaku torku, koja je ažurirana datom naredbom.

Sintaksa trigera u ORACLE ver. 7 je sledeća:

```
CREATE TRIGGER [<šema>.]<naziv_trigera>
      {BEFORE|AFTER} {[INSERT] [OR]
                     [DELETE] [OR]
                     [UPDATE [OF <lista_kolona>]]} ON
                         [<šema>.]<tabela>
      [REFERENCING OLD AS <staro_ime>
                   NEW AS <novo_ime>]
      [FOR EACH ROW]
      [WHEN <predikat>]
      <pl/sql/blok>
```

Putem trigera se mogu definisati specifična pravila poslovanja (integriteta). Međutim, trigeri nisu pravila ograničenja (kao mehanizam samog softvera za rukovanje bazama podataka) nego su programi. Znači, korisnik ih posebno kreira za date potrebe. Implementacija pravila integriteta kao sastavnog dela šeme baze podataka je realizovana u aktivnim sistemima baza podataka: POSTGRES i STARBURST, koji su opisani u odeljku 1.3.3. i 1.3.4.

1.3.2. FoxPro, Access

Pojavom personalnih računara na tržištu, pojavili su se i softveri za razvoj baza podataka tipa FoxPro, Access, Clipper, Clarion. Ovi softveri ne mogu da se podvedu pod termin *softveri za rukovanje bazama podataka*, jer ne ispunjavaju u potpunosti sve zahteve, koji se pred takve softvere postavljaju:

- ovi sistemi nemaju (ili samo delimično imaju) mehanizme za opis implementacione šeme baze podataka (konceptualne šeme, eksterne i interne šeme) - rečnik podataka;
- ne obezbeđuju u potpunosti konkurentan rad za više korisnika (očuvanje konzistentnosti podataka prilikom upita ili provođenja promena);
- ne obezbeđuju transakcioni rad (obezbeđenje integriteta baze podataka);
- nemaju mehanizme za zaštitu od nenormalnih završetaka transakcija i automatsku rekonstrukciju podataka i očuvanje integriteta baze;

- delimično podržavaju upit nad distribuiranim bazama, istovremeni upit nad bazom od strane više korisnika u mreži.

Oni su našli svoju primenu u razvoju aplikacija manjeg obima, složenosti i zahteva. Pogodni su kao alat za obuku početnika u rukovanju softverima za baze podataka.

FoxPro softver je proizvela kompanija Ashton Tate. Prvi proizvod ove kompanije bio je dBaseII 1981. godine i bio je proizveden za IBM PC. Zasnovan je na relacionom modelu podataka. Daljim usavršavanjem ovog softvera nastao je FoxBase. I drugi proizvođači su svojim proizvodima davali imena koja su sadržavala izraz *BASE* (rBASE i SuperBase), pa je *dbase* postalo ime, koje je najčešće korišćeno za softvere za baze podataka namenjene personalnim računarima. Svi proizvodi kompanije Ashton Tate za baze podataka zadržali su sintaksu i format datoteka dBASE II. Daljim usavršavanjem nastao je FoxPro 1.0. Kompanija Microsoft otkupila je 1992. godine kompaniju Ashton Tate i od tada se FoxPro razvija i pod Windows okruženjem, razvijaju se alati za izradu biblioteka u C jeziku, vrši se povezivanje sa SQL serverima i drugim platformama baza podataka, razvijaju se alati samostalnih izvršnih verzija. Verzija Visual FoxPro se pojavila 1994. godine i proširena je mogućnostima za rad sa objektima i za rad u grafičkom Windows okruženju. U ovoj verziji razvijen je mehanizam za očuvanje referencijalnog inegriteta putem definisanja tipa veze među tabelama u okruženju *Database Designer*. Moguće je definisanje kaskadnog i restriktivnog održavanja referencijalnog integriteta.

Evolucija FoxPro okruženja je tekla u pravcu povećanja broja istovremeno otvorenih datoteka, pouzdanije strukture baza podataka, automatskog upoređivanja i kontrole u toku unošenja podataka, uvođenja alata (generatori maski, izveštaja, menija), razvoj osnovnih mogućnosti za rad u višekorisničkom okruženju (zaključavanje datoteka i mogućnost pristupanja datotekama u deljenom režimu), dodavanjem novih tipova podataka (memo polja, OLE - objektno povezivanje i ugrađivanje), kreiranjem strukturiranih i složenih indeksa, ugrađivanjem SQL sintakse.

ACCESS je prozvod kompanije Microsoft. Prva verzija se pojavila 1992. godine i to je bio prvi softver za razvoj baza podataka u Windows okruženju. On je zasnovan na relacionom modelu podataka. Ovaj softver se po nekim osobinama približava softverima za rukovanje bazama podataka, ali se one nalaze tek u početnim fazama razvoja. Podržava dinamički skup slogova (*dynaset*), ima skroman rečnik podataka, ima mehanizme za održavanje referencijalnih integriteta (uključuje podršku za kaskadno brisanje i ažuriranje) i ima SQL upitni jezik. Access je u osnovi neproceduralno okruženje, mada ima u osnovi proceduralan jezik *Visual Basic*. Access ima mogućnost manipulisanja podacima korišćenjem Visual Basic koda putem modula: *Data Access Objects* (DAO). Ima podršku za upravljanje događajima (event-driven procedure); podržava višekorisničko deljenje baze podataka i u file-server i u klijent-server okruženju; poseduje podršku za replikaciju baze podataka.

1.3.3. POSTGRES (DATALOG i POSTQUEL)

POSTGRES je razvijen na Kalifornijskom univerzitetu pod rukovodstvom prof. dr M. Stonebrakera. Ovaj sistem razvijen je kao ekstenzija nad INGRES relacionim softverom za rukovanje bazama podataka. POSTGRES je ***prototipski sistem***. POSTGRES-ov model podataka koristi relacionu tehnologiju, što znači da se baza podataka sastoji od skupa vremenski promenljivih relacija. Upitni jezik zasnovan je na relacionoj algebri i relacionom računu. Relacioni sistem POSTGRES-a proširen je mogućnostima za upravljanje znanjem i upravljanje objektima.

Prikaz POSTGRES-a dat je na osnovu [SRH90], [Ma96], [Ma93], [FT95].

POSTGRES kao relacioni sistem. POSTGRES-ov upitni jezik je POSTQUEL i predstavlja uopšteniju verziju QUEL-a. Poboljšanja u POSTQUEL-u su [SRH90]:

- uvođenje ***from*** klauzule, čime je omogućeno definisanje promenljive tipa *red* (n-torka),
- mogućnost pisanja izraza kojim se formira relacija na bilo kojem mestu u upitu,
- uvođenje u upitni jezik *execute* (izvrši) naredbe i naredbe za iteraciju upita,
- mogućnost referenciranja na podatke iz prošlosti.

Najjednostavniji oblik upita u POSTQUEL-u je spoj QUEL upitnog izraza (*retrieve* naredba) i SQL upitnog izraza (*select* naredba):

retrieve <lista atributa>
from <lista relacija>
where <uslovi selekcije ili spoja>

Za razliku od QUEL-a, POSTQUEL ima **from** klauzulu da bi se definisale promenljive n-torke (promenljive tipa ***red***). Opšta sintaksa **from** klauzule ima sledeći izgled:

from t_1 in R_1,...,t_n in R_n.

Značenje ***retrieve***-naredbe POSTQUEL-a je identično značenju sledeće ***retrieve***-naredbe QUEL-a:

range of t_1 is R_1
...
range of t_n is R_n
retrieve <lista atributa>
where <uslovi selekcije i/ili spajanja>

odnosno značenju sledeće ***select***-naredbe SQL-a:

select <lista atributa>
from R_1 t_1,...,R_n t_n
where <uslovi selekcije i/ili spajanja>

Domeni obeležja mogu biti predefinisani (standardni) i korisnički definisani.

Naredbe jezika za opis podataka i jezika za manipulaciju podacima POSTQUEL-a su: CREATEDB, DEFINE VIEW, DEFINE INDEX, RETRIEVE, APPEND, REPLACE, DELETE.

Upravljanje objektima je postignuto ekstenzijom relacionog modela. Ta ekstenzija se sastoji iz:

- definisanja apstraktnih tipova podataka (uključujući korisnički definisane operatore i procedure, atribute relacija tipa *procedura*)
- mogućnost nasleđivanja atributa i procedura.

Realizovani su sledeći koncepti: agregacija, generalizacija, složeni objekti i atributi koji referenciraju n-torke drugih relacija.

POSTGRES se svrstava u ***proširene relacione sisteme***. Može se definisati i kao ***objektno - orijentisani*** sistem, jer dodeljuje jedinstveni identifikator svakom objektu, odnosno, svakoj n-torki baze podataka, omogućuje definisanje apstraktnih tipova podataka, klasa i metoda (funkcija) i podržava nasleđivanje podataka i funkcija.

U POSTGRES-u su izjednačeni koncepti *relacije* i *klase*, s jedne strane, i *n-torke* i *primerka* (pojavljivanja) klase, s druge strane.

Svaka relacija (klasa) ima i *surogat ključa* - virtuelnu kolonu (atribut), koja jedinstveno identifikuje svaku n-torku relacije (svaki primerak klase). To znači da je u ovom softveru putem surogata realizovan integritet objekta.

POSTGRES ima mogućnost čuvanja izbrisanih ili izmenjenih podataka, jer on čuva sve podatke. Brisanje nepotrebnih podataka vrši se na eksplicitan zahtev korisnika.

Mogućnost ***iteracije upita***. Iteracija upita označava se dodavanjem simbola zvezdice kao sufiksa na POSTQUEL naredbe za manipulisanje podacima. Retrieve* naredba nastavlja da se izvršava sve dok nijedna n-torka ne ostane za prikaz, a naredbe append*, delete* i replace* sve dok nijedna n-torka ne ostane za ažuriranje.

Primer 1.8. Data je šema relacije *Radnik*({MBR, IME, PREZIME, RUKOVODILAC}, {MBR}), pri čemu obeležje *rukovodilac* označava ime neposrednog rukovodioca datog

radnika. Putem sledećeg upita kreira se relacija, koja sadrži sve radnike koji su *direktno* ili *indirektno* podređeni radniku Ivanu:

```
retrieve* into PODREĐENI(R.ime, R.rukovodilac)
from R in RADNIK, P in PODREĐENI
where R.ime = "Ivan"
      or R.rukovodilac = P.ime
```

POSTGRES kao aktivni sistem baza podataka

Osnovu aktivnih sistema čine sledeće opcije:

- prepoznati definisane situacije (u bazi podataka, aplikacijama, okruženju)
- pokrenuti definisane (re)akcije (operacije nad bazom podataka ili programe) po nastanku situacija.

Aktivni sistemi se odlikuju svojim **ECA (Event-Condition-Action) pravilima**. Pravila čine model znanja aktivnog sistema. Pored logičke šeme i činjenica, aktivni sistemi omogućuju i dodatnu semantiku putem složenih pravila integriteta i pravila ponašanja. ***Pravila su sastavni deo šeme baze podataka***, i zbog toga ovi sistemi imaju veću semantičku moć u odnosu na klasične relacione sisteme (ORACLE, INGRES...).

Događaj može biti:

- početak ili završetak operacije nad bazom podataka,
- početak ili završetak imenovane transakcije,
- vremenski događaj,
- istorijski događaj (praćenje istorijata neke pojave kroz frekvenciju njenog pojavljivanja u vremenskom intervalu),
- apstraktan događaj (pritisak na funkcijski taster).

Upravljanje znanjem realizovano je putem jedinstvenog opštenamenskog sistema pravila. Sistem pravila je integritetna komponenta POSTGRES-a i koristi se za podršku referencijalnog i poslovnog integriteta, pogleda, memorisanih procedura, zaštite, verzija, za programiranje rukovođeno događajima.

Prvi POSTGRES-ov sistem pravila zasniva se na ideji da se bilo koja POSTGRES-ova naredba može transformisati u pravilo promenom semantike naredbe, tako da se naredba logički izvršava ili ***uvek*** (ALWAYS) ili ***nikad*** (REFUSE) ili ***tačno jedanput*** (ONE-TIME) sa ispunjenjem definisanih uslova. Sintaksa ovog skupa pravila izgleda:

```
{ALWAYS | REFUSE | ONE-TIME} POSTQUEL-naredba
```

Primer 1.9. Data je šema relacije *Učenik*({MBR, IME, PREZIME, PROS_OC}, {MBR}), pri čemu obeležje *pros_oc* predstavlja prosečnu ocenu datog učenika. Sledećim upitom se prikazuje prosečna ocena učenika Mirka nakon 20. juna:

```
ONE-TIME retrieve (UČENIK.pros_oc)
where UČENIK.ime="Mirko" and TIME() > "Jun 20"
```

Prvi skup pravila POSTGRES-a je rekord (tuple) orijentisan: pokreće ih promena nad pojedinačnom n-torkom relacije.

Drugi POSTGRES-ov sistem pravila predstavlja način predstavljanja znanja u vidu produkcionih, ECA (Event-Condition-Action) pravila i ima izgled:

define [**rewrite**] **rule** <naziv>
on <događaj> **to** <objekat>
[[**from** <lista relacija>] **where** <uslov>]
then do [**instead**] <lista akcija>

pri čemu je:

- <događaj> neka od standardnih POSTQUEL naredbi za manipulisanje podacima: **append, delete, replace** ili **retrieve**, **new** sa značenjem *append* ili *replace* i **old** sa značenjem *delete* ili *replace*,
- <objekat> je naziv relacije ili lista naziva atributa relacija,
- **from** i **where** klauzule su standardne POSTQUEL klauzule,
- <lista akcija> je lista koja se sastoji od bilo koje akcije nad bazom podataka sa izmenom, koja omogućuje upotrebu **new** ili **current** ključne reči umesto promenljive tipa red.

Pravila iz drugog skupa POSTGRES-ovih pravila su, takođe, rekord-orijentisana, a to znači da ih pokreću promene nad n-torkama relacija.

Semantika pravila određuje da je u trenutku pristupa, ubacivanja, izmene ili izbacivanja n-torke poznata **current** (**tekuća**) n-torka (za prikaz, izmenu, izbacivanje) i **new** (**nova**) n-torka (za izmenu i ubacivanje). Ako je nakon nastanka događaja, za tekuću n-torku zadovoljen uslov iz **where** klauzule, pokreće se izvršavanje akcija pravila. Nakon što se akcija završi, atributi koji su referencirani sa **current**.<naziv_atributa> i **new**.<naziv_atributa> dobijaju vrednosti odgovarajućih polja tekuće, odnosno nove n-torke.

Primer 1.10. Data je šema relacije:
Razred({SIF_RAZ, SIF_RAZ_ST, BROJ_UČENIKA},{SIF_RAZ, SIF_RAZ_ST})

gde obeležje *br_učenika* ima značenje: broj učenika datog razreda, a obeležje *sif_raz_st* ima značenje: šifra razrednog starešine. Putem sledećeg pravila definisana je akcija - prikaz šifre razrednog starešine i šifre razreda, koga pohađa manje od 10 učenika (pravilo se okida kada se taj broj učenika promeni):

define rule Prikazi_podatke_razred
on replace **to** razred.broj_učenika
where razred.broj_učenika<10
then do retrieve razred.šif_raz_st, razred.šif_raz

U pravilu je moguće definisati da se promena (operacija), koja je prouzrokovala pokretanje pravila, uopšte ne izvrši. To se postiže klauzulama:
refuse - promena, koja je inicirala pokretanje pravila, se odbija,
instead - umesto promene izvršava se skup operacija definisanih **instead** klauzulom.

POSTGRES kao deduktivni sistem baza podataka

Deduktivni sistemi baza podataka koriste model podataka koji koristi relacionu notaciju podataka i predikatski račun prvog reda za predstavljanje znanja i definisanje dozvoljenih operacija nad podacima. Ovaj model podataka je implementiran u POSTGRES-u pod imenom **DATALOG**. Naziv ovog modela podataka (tačnije rečeno paradigme logičkog programiranja u oblasti baza podataka) potiče od osnovne namene: predložiti verziju PROLOG-a, koja je podesna za sisteme za rukovanje bazama podataka.

Matematički model podataka, na kome je razvijen DATALOG, istovetan je konceptima na kojima se zasniva relacioni model. Predikatski simboli u DATALOG-u označavaju relacije. Referenciranje atributa relacije ostvaruje se po poziciji argumenata, jer se oni navode u fiksnom redosledu. Na primer, ako je ***p*** predikatski simbol, onda u predikatu **p(X,Y,Z)** promenljiva X označava prvi atribut (kolonu) relacije, koja odgovara predikatu **p**.

Karakteristike DATALOG-a, kao sistema deduktivnih baza podataka, opisane su u **odeljku 3.3. *DATALOG i CWA-princip***. U [Ullm88] i [Ma93] opisan je način realizacije DATALOG-a u upitnom jeziku POSTGRES-a: POSTQUEL-u. Pri tome, osnovna namera je da se u POSTQUEL-u podrži rekurzija i realizuje iteracija upita. U POSTQUEL-u se bilo koje nerekurzivno DATALOG-pravilo predstavlja izrazom bez iteracije, a bilo koje rekurzivno DATALOG-pravilo predstavlja se posebnim *retrieve** iterativnim izrazom. Pri tome se uvažavaju ograničenja u odnosu na specifičan oblik pravila (Hornove klauzule).

Na taj način razvijen je mehanizam koji putem iteracije upita obezbeđuje mogućnost izvođenja novog na osnovu postojećeg znanja, a to je najznačajnija karakteristika deduktivnih sistema.

1.3.4. STARBURST

STARBURST je prototipski relacioni sistem koji se razvija u IBM-ovoj istraživačkoj laboratoriji u Kaliforniji od 1985. godine.

STARBURST softver za rukovanje bazama podataka je zasnovan na relacionom modelu. Međutim, njegova najvažnija karakteristika je ***mogućnost proširenja***. Proširenje sistema se odnosi na sledeće aspekte:

- podrška novih operacija,
- prilagođavanje upravljanja podacima,
- podrška za aplikacije.

Podrška novih operacija obuhvata dodavanje novih operacija u cilju usavršavanja upitnog jezika. Upitni jezik STARBURST-a je **Hydrogen**, proširena verzija SQL-a. Svaka promena upitnog jezika zahteva promene u modelu grafa upita, određivanje semantičke optimizacije upita, dodavanje novih strategija izvršavanja, omogućavanje izvršavanja novih operacija i definisanje privilegija.

Prilagođavanje upravljanja podacima obuhvata nove načine memorisanja podataka i izmene jezika za definiciju podataka. Za ove operacije potrebno je da postoji dobro definisan skup izvršnih operacija (scan, fetch, insert, update, delete, drop table).

Podrška za aplikacije obuhvata definisanje novih tipova podataka i složenih objekata, podršku novim funkcijama sortiranja i izgradnju sistema pravila. Putem pravila omogućava se automatsko izvršavanje određenih operacija kada se desi neki događaj i ako su ispunjeni određeni uslovi.

Sintaksa pravila je sintaksa SQL-a, s obzirom da je Hydrogen, upitni jezik STARBURST-a, proširenje u odnosu na SQL. Relacioni upitni jezici su *skupovno orijentisani*, (odnose se na skup torki) pa su i pravila *skupovno orijentisana* (pravila pokreću promene nad *skupom* torki, a akcije u pravilima se, takođe, odnose na *skup* torki). Pravila su zasnovana na ***prelazu stanja***. Prelaz stanja je nedeljiva celina operacija za ažuriranje baza podataka (insert, update, delete). Pravilo u STARBURST-u ima oblik:

create rule <naziv_pravila> **on** <tabela>
when <predikat_prelaza>
[**if** <uslov>]
then <lista_akcija>
[**precedes** <lista_pravila>]
[**follows** <lista_pravila>]

Predikat prelaza određuje operacije koje pokreću izvršavanje pravila. To su sledeće operacije: **inserted**, **deleted**, **updated**.

Uslov je proizvoljan SQL predikat nad bazom podataka. Ako je uslov zadovoljen, izvršavaju se akcije, koje su navedene u listi akcija. Ako uslov nije naveden, smatra se da se akcije pravila izvršavaju bezuslovno, odnosno uvek kad god se pokrene pravilo.
Akcija je bilo koja STARBURST operacija nad bazom podataka:

- operacija za manipulisanje podacima (**insert**, **delete**, **update**, **select**)
- operacija za kontrolne funkcije (**rollback**),
- operacija za definisanje podataka (**create table**).

Neobavezne klauzule **precedes** i **follows** definišu redosled primene, odnosno prioritet pravila.

Primer 1.11. Date su sledeće šeme relacija:
Nastavnik({SIF_NAST,IME,PREZIME},{SIF_NAST})
Predmet({SIF_PRED,NAZIV},{SIF_PRED})
Predaje({SIF_PRED,SIF_NAST},{SIF_PRED,SIF_NAST})

Definisati pravilo kojim će se obezbediti kaskadno održavanje referencijalnog integriteta između šema relacija *Nastavnik* i *Predaje*; kada se brišu podaci o nekom nastavniku, potrebno je putem pravila definisati akciju brisanja odgovarajućih podataka u relaciji *Predaje* (podatke da nastavnik, čiji se podaci brišu iz baze, predaje neke predmete).

```
create rule cascade on nastavnik
when deleted
then delete from predaje
        where predaje.sif_nast in
        (select sif_nast from deleted)
```

Sistem pravila STARBURST-sistema služi za očuvanje integriteta baze podataka, kao sredstvo za održavanje materijalizovanih pogleda, realizaciju deduktivnih baza podataka, kontrolu prava pristupa (autorizacije), održavanje verzija, što znači da predstavlja kompleksan i moćan mehanizam aktivnog softvera za rukovanje bazama podataka.

2. BAZE PODATAKA U DIDAKTIČKOJ TRANSFORMACIJI ZNANJA

U ovom poglavlju izložene su teoretsko-didaktičke osnove projektovanja obrazovnog računarskog softvera. Dat je prikaz primene modela podataka u projektovanju baza podataka za pojedine tipove obrazovnog softvera u odnosu na njihove karakteristike. Opisani su karakteristični postupci projektovanja baza podataka putem relacionog, objektno-orijentisanog, logičkog i kombinacijom relacionog i logičkog modela.

Sadržinu ovog poglavlja čine sledeći odeljci:

- didaktička transformacija znanja
- projekcija mogućnosti i rešenja u domenu baza podataka na ciljeve i zadatke obrazovnog računarskog softvera,
- korišćenje modela podataka za realizaciju pojedinih elemenata koji mogu biti prisutni u raznim tipovima obrazovnog računarskog softvra.

2.1. Pojam didaktičke transformacije znanja

U ovom odeljku se na osnovu literature izdvajaju najbitniji prilazi problemu usvajanja znanja u oblasti nastave i učenja i ukazuje se na mogućnosti za transformaciju i povezivanje raznih oblika predstavljanja znanja.

Po Felix von Cube-u ***didaktika*** je nauka o razvoju i optimizaciji strategija poučavanja. Ona se sastoji od ***metoda*** i realizuje se putem ***sredstava***. "***Kibernetička didaktika*** sastoji se u primeni kibernetičkih pojmova i metoda (regulacioni krug, teorija informacija i sl.) na predmetno područje vaspitanja ili obrazovanja." [Cu91]

Upravljanje u nastavi

Najvažniji zadatak teorije učenja i nastave je što više povećati punoću upravljanja. Stav didaktičke teorije je da je kod učenika potrebno formirati ne samo determinisane već i nedeterminisane (tačnije, nepotpuno determinisane) procese. [La75].

Upravljanje u nastavi može biti **direktno** ako se učeniku daju uputstva putem kojih učenik uči (rešava zadatak). Upravljanje nije direktno ako se formiranje operacije ostvaruje pomoću određene organizacije sadržaja nastavnog gradiva, stvaranjem uslova da se **sam učenik** suoči sa tom operacijom. I zato se, po didaktičkoj teoriji, smatra da upravljanje u procesu učenja može biti **potpuno** i **nepotpuno**, odnosno **čvrsto** i **labavo**.

"Strategija poučavanja sastoji se u sledu planiranih mera koje treba da sprovedu nastavnik ili mediji, i koje adresata treba da vode određenom cilju poučavanja". [Cu91]

Strategije učenja mogu biti [La75]:

- od striktnog (čvrstog) upravljanja ka nestriktnom
- od nestriktnog ka striktnom.

Određivanje vrste strategije zavisi od:
a) ciljeva nastave,
b) psiholoških mogućnosti i stanja učenika.

Modeli i sistemi nastave i modeli učenja

U didaktičkoj teoriji se smatra "da se modeli učenja i modeli nastave mogu koristiti kao sinonimi ako se modeli učenja nalaze inkorporirani u modelima nastave i sa njima čine jedinstvo i sklad. Modeli nastave, međutim, po svojoj strukturi su širi, jer zahtevaju pored metoda učenja i *ciljeve, strategije i sadržaje učenja*." [Vos95]

Klasifikacije modela nastave

Sistemi nastave po Đ. Lekiću

Đ. Lekić u [Le76] navodi sledeće sisteme nastave:

- sistem poučavanja,
- samostalan rad,
- sistem programiranja nastave.

Kod **sistema poučavanja** nastava je precizno teorijski definisana i podrazumeva se naučno-zasnovan vaspitno-obrazovni rad. Ovaj sistem nastave je, u stvari, klasičan didaktički sistem poučavanja. On ima nedostatke koji se ogledaju u tome što se nastava ne može prilagoditi psihofizičkim odlikama pojedinaca, a u didaktičkoj praksi je poznato da nastava bez moguće individualizacije nema puni vaspitno - obrazovni efekat. Prednosti se ogledaju u mogućnosti rada sa većim brojem učenika i u precizno definisanim ciljevima nastave što ima za rezultat racionalizaciju nastave. Nastava je usmerena ka realizaciji osnovnih zadataka.

Samostalan rad u nastavi je pogodan za razvoj i podsticanje učeničkih konativnih sposobnosti jer podstiče voljno - emotivne reakcije učenika. Nedostatak ovog sistema nastave je što se zanemaruju učeničke kognitivne sposobnosti kao posledica zanemarivanja usmerenih intelektualnih napora učenika. Ovaj sistem je pogodan za razvoj

učenikovih individualnih kreativnih sposobnosti jer nije ograničen didaktičkim ciljevima i zadacima. Omogućena je individualizacija nastavnog procesa. Upravo zbog toga, ovaj sistem nastave nije pogodan (ekonomičan) za veće grupe učenika, gde se zahteva organizacija nastavnog procesa. Tok nastave nije usmeren ka jasno definisanom cilju pa se zbog toga može desiti da se vreme i energija za učenje troše u većem obimu nego što je to slučaj kod sistema poučavanja.

Sistem programirane nastave je sistem gde je nastavno gradivo najvažnije u čitavom nastavnom procesu. I nastavnik i učenik su orijentisani ka nastavnom gradivu, ali se i nastavnik i učenik shvataju, takođe, kao relevantni faktori nastavnog procesa. "Nastavnik neposredno programira (priprema, oblikuje, sređuje, umnožava) nastavno gradivo, a učenik neposredno uči (obrađuje, utvrđuje, ponavlja, vežba i vrednuje) radi savladivanja i usvajanja." ([Le76], str.30) U ovom sistemu nastave se želi da se isticanjem gradiva, kao neutralnog faktora nastavnog procesa, eliminišu mogući subjektivni odnosi između učenika i nastavnika.
Da bi nastavnik uspeo u primeni programirane nastave on mora da poznaje [Le76]:

- reprezentativnost sadržaja,
- određenost zadataka,
- sistematičnost postupanja,
- odmerenost ritma,
- pravovremenost saznanja rezultata.

Modeli nastave po N. Landi

Autor N. Landa razlikuje sledeće modele nastave:

- algoritamski
- heuristički modeli
- polualgoritamski i poluheuristički modeli.

Za ovu podelu modela nastave karakteristično je što strogost (determinisanost) upravljanja nastavnim procesom opada idući od algoritamskih ka heurističkim modelima.

Ovaj autor navodi rezultate didaktičkih istraživanja, koja su vršena u oblasti rešavanja geometrijskih zadataka i učenja gramatike. U ovim istraživanjima je izvršeno upoređivanje metoda (uputstava) za rešavanje datih zadataka. Uputstvo za rešavanje gramatičkih zadataka je precizno određivalo radnje učenika (to su bili algoritmi), dok je uputstvo za rešavanje geometrijskih zadataka sadržavalo uputstva, koja nisu u potpunosti određivala učenikove radnje (odnosno ta uputstva su bila heurističkog tipa). Na osnovu ovih istraživanja uočilo se da algoritmi garantuju rešenje zadatka (ovo je uočeno na primeru rešavanja gramatičkih zadataka), dok heurističke metode ne garantuju rešenje, ali se u mnogim slučajevima dolazi do cilja sa manjim utroškom vremena i sredstava.

Naime, osnovno svojstvo algoritma je da on u potpunosti i jednoznačno determiniše rešavanje zadataka od strane čoveka i da njegova primena ne zahteva stvaralaštvo, iako je
za otkrivanje i stvaranje algoritma stvaralaštvo potrebno. Zbog toga se u didaktici i smatra da učenicima nije potrebno saopštavati algoritme u gotovom obliku, nego da učenici *samostalno* pod rukovodstvom nastavnika kreiraju algoritme. Na taj način se u modelima nastave vrši integracija algoritamskog i heurističkog modela učenja.

Galjperinova teorija etapnog formiranja umnih radnji

Po ovoj teoriji se proces formiranja umnih radnji sastoji od:

- etape formiranja pojedine radnje (prethodno upoznavanje sa radnjom),
- etapa materijalne radnje,
- etapa glasnog govora,
- etapa glasnog govora u sebi ("unutrašnjeg govora")
- etapa umnog rada.

Ova teorija tvrdi da je mogućno nepogrešivo obučavanje ili učenje i da je potrebno pomoću uputstava u potpunosti upravljati misaonim procesima.

"Međutim, u nauci je dokazano (Gedelova teorema) da postoje klase zadataka za koje je principijelno nemoguće sastaviti opšta i jednoznačna uputstva, koja bi omogućila da se zadaci date klase rešavaju na jedinstven način." [La75]. Zato se u didaktičkoj teoriji smatra da je nemoguće potpuno determinisati tok rešavanja svih tipova zadataka. Galjperinova teorija ima samo relativnu vrednost, jer se ograničava samo na jedan tip zadataka i samo jedan tip procesa njihovog rešavanja (algoritamski).

Izbor modela nastave

Stav didaktičke teorije je da izbor modela nastave zavisi od [Vos95]:

- nastavnih ciljeva,
- prirode nastavnih sadržaja i
- individualnih karakteristika učenika.

Ako je potrebno da se brzo i potpuno usvoje jasno definisane, precizno određene informacije na osnovu davanja jednoznačnih uputstava učeniku onda se koriste algoritamski modeli. Ovaj model odgovara učenicima sa prosečnom inteligencijom, koji nemaju samopouzdanja.

Ako je cilj nastave da se razviju kreativnost i sposobnost apstraktnog mišljenja onda se koriste heuristički modeli. Oni su pogodni za modelovanje gradiva koje ima složenu

logičku strukturu. Ovaj model odgovara učenicima sa razvijenijom inteligencijom, koji imaju više samopouzdanja i koji su nezavisni.

Polualgoritamski i poluheuristički modeli učenja se koriste ako se želi transformacija poučavanja i instrukcije kompjutera u samoučenje i samoinstrukciju učenika [Vos95]. U ovim modelima primena didaktičkih softvera dolazi do punog izražaja. Počev od 70-ih godina u didaktici se sve veća pažnja poklanja poluheurističkim i heurističkim modelima učenja i nastave. Upotrebom didaktičkih softvera realizuju se aktivnije i fleksibilnije strategije učenja. Učenici samostalno otkrivaju rešavanje zadataka.

U didaktičkoj teoriji [Kv78] se smatra da je primenom didaktičkih softvera u nastavi omogućena individualizacija nastave "pružanjem mogućnosti da učenici sami biraju modele učenja, individualizacijom obima i složenosti nastavnog gradiva, načinom i tempom njegove obrade, primenom tehnike "preskakanja", uvažavanjem različitih sposobnosti učenika (tipovi mišljenja) i osobina ličnosti (motivacija, karakter, temperament, anksioznost, neurotičnost i dr.)"

Međutim, smatra se, takođe, da je sva tri modela nastave (algoritamski, heuristički i poluheuristički) potrebno koristiti u razvoju didaktičkog softvera. Izbor poluheurističkog ili heurističkog modela učenja određen je ciljevima nastave, sadržajima učenja i psihološkim mogućnostima učenika. U modelima i sistemima nastave i učenja, strategijama učenja i upravljanju u nastavi važnu komponentu predstavlja mogućnost ***didaktičke transformacije znanja***.

Felix von Cube u [Cu91] kaže: "Primena informacijskoteorijskih metoda u didaktici doprinosi preciziranju i optimiranju strategija poučavanja: tako se npr. može navesti kako tekst treba strukturirati da bi za učenika imao najveću informativnu vrednost; može se izračunati broj ponavljanja neophodnih za "skladištenje", kao što se može navesti da li se informacija superznacima može, i kako, reducirati...
...U novije doba se teorija informacija (u sprezi sa teorijom znakova) primenjuje i u didaktici nastavnih sredstava. Pritom se pokušavaju precizirati i optimirati uobičajena kodiranja (npr. kodiranje stvarnosti posredstvom audio-vizualnih sredstava, kodiranje apstraktnih sklopova digitalnim sredstvima ili šemama). Time se povisuje dejstvo učenja u kodiranim strategijama poučavanja."

Može se reći da je didaktička transformacija znanja jedan vid heurističkog modela učenja. Sam pojam ***didaktičke transformacije*** znanja se može posmatrati na ***makro i mikro planu.*** Na ***makro planu*** pod didaktičkom transformacijom podrazumeva se transformiranje određenog znanja u oblik koji je didaktički prihvatljiv. Na primer, određene naučne činjenice moraju se transformisati u skladu sa ciljevima, mogućnostima učenika i drugim uslovima u kojima se znanje usvaja odnosno ne mogu biti prezentovane u izvornom obliku. Na ***mikro planu*** didaktička transformacija podrazumeva razne oblike predstavljanja znanja i njihovo povezivanje, odnosno transformaciju jednog oblika

predstavljanja znanja u drugi oblik. Na primer, ako se omogući predstavljanje datog pojma u obliku teksta ili slike ili animacije ili putem nekog drugog analognog pojma (koji ima izvesnu zajedničku karakteristiku sa datim pojmom) onda se učeniku omogućava da sam bira puteve i načine rešavanja zadataka čime se doprinosi individualizaciji nastave.

Cilj didaktičke transformacije znanja je da učeniku omogući efikasnije:
- učenje postojećeg znanja - učenik treba da razume i usvoji određena znanja,
- sticanje novog znanja otkrivanjem - učenik treba ***sam*** da otkrije postojeća znanja.

To se odnosi kako na usvajanje pojmova, postupaka, činjenica i veza među njima, tako i na sam proces rešavanja zadataka (u najširem smislu). Pri tome, osnovnu ulogu imaju:
- oblici predstavljanja znanja (problema) i
- izbor efikasne strategije (rešavanja problema).

Oblici predstavljanja znanja bitno utiču na efikasnost rešavanja zadataka i na efikasnost samog procesa učenja. "Od izbora oblika predstavljanja znanja zavisi složenost opisa zadataka, a samim tim i efikasnost njegovog rešavanja." ([Hot95], str. 17) Takođe je u [Hot95] pokazano kako se jedan oblik predstavljanja znanja transformiše u drugi adekvatniji oblik, koji omogućuje ne samo efikasnije rešenje postojećeg problema, nego omogućuje njegovu generalizaciju i sticanje novog opštijeg znanja.

Izbor adekvatnog oblika predstavljanja znanja vrši se u skladu sa prirodom samih sadržaja ali i u skladu sa individualnim svojstvima onog ko treba da usvoji te sadržaje. Za takav izbor neophodno je obezbediti mogućnost transformacije jednog oblika predstavljanja znanja u drugi oblik. "Nažalost, za problem transformacije predstavljanja još ne postoji opšte teorijsko rešenje. Izbor oblika predstavljanja i njegovo transformisanje prepušteni su kreatoru. U osnovi radi se o izboru takvog oblika predstavljanja koji je usklađen sa sredstvima koja pomažu pri učenju..." ([Hot95], str. 20) To znači da se odluka o izboru oblika prestavljanja znanja i o transformacijama u druge oblike, donosi heurističkim metodama i zavisi od ciljeva, sredstava i drugih raspoloživih mogućnosti.

S obzirom da je ovo istraživanje usmereno na projektovanje baza podataka za potrebe obrazovnog računarskog softvera, u narednim odeljcima se pokazuje kako se na sadašnjem nivou razvoja metoda i tehnika u projektovanju i rukovanju bazama podataka može ostvariti didaktička transformacija znanja.

Uzimajući u obzir prethodno iznete stavove, može se reći da je bazu podataka, koja se koristi za potrebe didaktičkog (obrazovnog) softvera, potrebno projektovati tako da se omogući odgovarajuća didaktička transformacija znanja, odnosno, adekvatni oblik

predstavljanja znanja. Ovaj zadatak se svodi na izbor adekvatnog modela podataka i izbor tehnika koje su sadržane u softverima za rukovanje bazama podataka. Model podataka putem svojih formalno - apstraktnih koncepata treba da predstavi činjenice, znanja, probleme, zadatke iz odgovarajuće predmetne oblasti i da realizuje njihovu povezanost. Softver za rukovanje bazama podataka treba da omogući izbor određene strategije korišćenja sadržaja baze, što uključuje: realizaciju upita, ažuriranje podataka u bazi, višekorisnički rad, kontrolu i zaštitu pristupa podacima u bazi, transakcioni rad, efikasno korišćenje baze podataka i upravljanje distribuiranim delovima baze podataka.

2.2. Projekcija rezultata iz domena baza na ciljeve obrazovnog računarskog softvera

U poglavlju 1. navedene su osnovne karakteristike savremenih modela podataka: relacionog, objektno - orijentisanog i logičkog. Takođe je dat prikaz savremenih pravaca razvoja u bazama podataka koji se odnose na razvoj ***aktivnih baza podataka***. U ovom poglavlju izvršena je analiza modela podataka i na njima zasnovanih softvera za rukovanje bazama podataka u odnosu na mogućnosti njihove primene u postupcima projektovanja i korišćenja različitih tipova obrazovnog računarskog softvera. Klasifikacija tipova obrazovnog računarskog softvera se nalazi u literaturi [Nad94].

Analiza je izvršena za svaki tip obrazovnog softvera u odnosu na sledeće karakteristike:

- u kojoj meri se cilj datog tipa softvera ostvaruje uz prisustvo određenih baza podataka i koje baze podataka su u pitanju (npr. da li se model učenika nalazi u svim tipovima softvera)
- koji model baza podataka je adekvatan u odnosu na vrstu softvera, odnosno njegove ***ciljeve, zadatke i specifičnosti***.

U [Nad94] je izvršena klasifikacija tipova obrazovnog softvera na osnovu različitih kriterijuma i od strane nekoliko autora. U ovom istraživanju će se uzeti klasifikacije koje su dali Đ. Nadrljanski i Tejlor, jer one većim delom obuhvataju i ostale klasifikacije.

2.2.1. Mogućnosti primene baza podataka za realizaciju ciljeva i zadataka obrazovnih softvera u odnosu na klasifikaciju po stepenu samostalnosti

Ova klasifikacija polazi sa stanovišta kibernetičkog prilaza. Suština te klasifikacije je ***princip samostalnosti u upravljanju procesom učenja od strane korisnika***. "Da bi korisnici svoje obrazovne i radne ciljeve ostvarili, potrebno je da sami oblikuju sadržaj programa i upravljaju sredstvima informatičke tehnologije. Stepen samostalnosti i

upravljanja korisnika u učenju je različit i kreće se od vođenog ili upravljanog procesa učenja pa do primene ekspertnog sistema." [Nad94]

Po ovoj klasifikaciji postoje sledeći tipovi softvera (pri čemu stepen samostalnosti upravljanja procesom učenja od strane korisnika raste idući od upravljačkog obrazovnog softvera ka ekspertnim sistemima):

- upravljački obrazovni softver,
- tutorski obrazovni softver,
- dijagnostički obrazovni softver,
- obrazovni softver za vežbanje,
- obrazovni softver tipa banke podataka (znanja),
- obrazovni softver tipa eksperimenta,
- obrazovni softver tipa simulacije,
- softverski alati,
- ekspertni sistemi.

Upravljački obrazovni softver

Karakteristike: sem elemenata obrazovnih sadržaja ovaj tip softvera ima i uputstvo za korišćenje koje daje korisniku sugestiju kad da prekine sa korišćenjem računara i da učenje nastavi nekom drugom metodom (udžbenik, eskperiment, konsultacije sa nastavnikom).

Ovaj tip softvera ima u svojoj bazi podataka definisan model učenika koji sadrži informacije o stečenom nivou znanja učenika tokom učenja uz pomoć obrazovnog softvera. Na osnovu tih informacija se daje sugestija da li da se nastavi sa korišćenjem softvera ili ne.

Za projektovanje i implementaciju ovog tipa softvera predlažu se ***aktivni sistemi baza podataka***. U poglavlju 1. objašnjene su osnovne karakteristike ovih sistema. U ovim sistemima se u formi pravila definišu:

- događaj koji izaziva datu akciju,
- uslov pod kojim se akcija sprovodi i
- sama akcija.

Implementaciju ovog tipa obrazovnog softvera objasnićemo na primeru post-relacionog softvera za rukovanje bazama podataka sa aktivnom komponentom - POSTGRES, čiji je detaljniji opis dat u poglavlju 1.3.3.

Pojmovi iz date nastavne oblasti, teme ili jedinice nalaze se memorisani u obliku tabela, s obzirom da je POSTGRES post-relacioni sistem i on je zadržao koncept tabele iz relacionog modela podataka. Osim gradiva u bazi podataka nalazi se i model učenika.

Primer 2.1. Model učenika se nalazi u formi sledeće šeme relacije:
UČENIK({IDENT_BR, IME, PREZIME, DATUM, NASTAVNA_JEDINICA, BODOVI}, {IDENT_BR})
Obeležje **bodovi** ima značenje: osvojeni broj bodova za datu nastavnu jedinicu za određeni datum.

Na osnovu ove relacije definiše se pravilo na osnovu koga će obrazovni softver da prikaže ime učenika i broj bodova kad god taj broj bodova ne pređe (ili pređe) određenu ***granicu***. Tu granicu definiše nastavnik na osnovu vrste gradiva i ciljeva časa i ona služi nastavniku kao kriterijum za prelazak na neki drugi oblik učenja. Pravilo ima sledeći izgled (pojam pravila je detaljnije objašnjen u poglavlju 1.3.3.):

define rule Poruka_Nastavniku
on replace **to** učenik.bodovi
where učenik.bodovi < *granica*
then do retrieve učenik.ime, učenik.bodovi

Za projektovanje baze podataka upravljačkog obrazovnog softvera odabran je aktivni sistem baze podataka, koji putem pravila definisanih u samoj bazi podataka omogućava pokretanje adekvatnih akcija kojima se usmerava proces učenja.

Tutorski obrazovni softver

Karakteristike: ovaj tip je predviđen za učenje obrazovnih sadržaja koji se uče u vidu programiranih vežbi koje korisnik mora istim redom da savlada. On u sebi ima krute algoritme upravljanja.

Ovaj tip softvera ima bazu podataka u kojoj se nalaze obrazovni sadržaji. Ne ukazuje se potreba za definisanjem modela učenika s obzirom na to da se učenje odvija uvek istim redosledom po krutom algoritmu na koga korisnik može malo da utiče.
U didaktici se nastavno gradivo predstavlja u obliku sledeće strukture:

- nastavno područje,
- nastavna oblast,
- nastavna tema,
- nastavna jedinica i
- didaktička celina,

pri čemu je nastavno područje koncept na najvišem nivou složenosti a didaktička celina koncept na najnižem nivou. Predstavljanje ovako strukturiranog nastavnog gradiva moguće je pomoću relacionog modela podataka u kome se za predstavljanje svakog od pobrojanih didaktičkih koncepata nastavnog gradiva koristi koncept relacije. Prikaz potrebnih informacija učeniku (korisniku) se vrši na osnovu operatora relacione algebre:

selekcije, projekcije, spoja i drugih operatora relacione algebre nad relacijama, koje prezentuju nastavne teme, jedinice i druge didaktičke koncepte. Primer ovog oblika predstavljanja pojmova nastavnog gradiva dat je u **poglavlju 2.3.1. *Projektovanje baza podataka za slučaj dekompozicije pojmova*.**

Upravo zbog ovih karakteristika se za realizaciju ovog tipa softvera predlaže relacioni model podataka i na njemu zasnovani softveri za rukovanje bazom podataka.

Za praktičnu realizaciju ovog tipa obrazovnog softvera se mogu koristiti neki od savremenih softvera kao što su: Access, Visual FoxPro. U ovim softverima su osim standardnih tipova podataka (znakovni, numerički, datumski, memo) implementirani i tipovi podataka za memorisanje slika, zvuka, animacije. Ovi tipovi podataka su neophodni za predstavljanje različitih oblika nastavnih sadržaja.

Na primer, u Visual FoxPro softveru implementiran je binarni tip podataka - *general.* Ovaj tip podatka je sličan memo poljima, jer i on služi za smeštanje podataka promenljive dužine. Međutim, razlikuje se po tome što je sadržaj, koji je predstavljen putem ovog tipa podatka, formatiran prema jednoj specifičnoj tehnici zvanoj OLE tehnika (Object Linking and Embedding - Povezivanje i ugrađivanje objekata). Putem ove tehnike, podatke koji su predstavljeni putem tipa *general* može koristiti neki drugi softver koji, takođe, podržava OLE tehniku. Na primer, WAV datoteka, koja je kreirana u softveru za snimanje zvuka, sadrži podatke za generisanje melodije na osnovu kojih program za reprodukciju zvuka može tu melodiju da odsvira.

U Access softveru za rukovanje bazama podataka tip podatka *OLE Object* služi sa smeštanje slika, dijagrama, zvučnih i video zapisa. U ovom softveru je moguće ugrađivanje ili povezivanje objekata iz sledećih softvera: PaintBrush, Sound Recorder i Object Packager.

Dijagnostički obrazovni softver

Karakteristike: koristi se za proveru znanja, sposobnosti i spretnosti korisnika. Ovaj tip softvera daje povratne informacije korisniku.

U bazi podataka ovog tipa softvera potrebno je da postoji model učenika jer se na osnovu njega daju povratne informacije. Model učenika sadrži sledeće informacije:

- informacije opšteg tipa: ime, prezime, razred
- specifične informacije: ime nastavne jedinice, broj bodova osvojenih u prethodnim proverama znanja, broj bodova tekuće provere.

Za potrebe projektovanja baze podataka ovog tipa softvera predlaže se logički model podataka, jer je potrebno saopštiti informaciju korisniku (učeniku) o tome da li je i u kom stepenu savladao gradivo. U logičkom modelu podataka moguće je definisati pravila

putem kojih će na osnovu podataka u bazi o svakom učeniku obrazovni softver dati potrebnu poruku korisniku (nastavniku - učeniku).

Za realizaciju ovog tipa softvera adekvatan je BASELOG-sistem, detaljnije objašnjen u poglavlju 4. Pravila putem kojih se realizuje dijagnostička komponenta ovog tipa softvera moguće je u BASELOG-sistemu predstaviti putem sopstvenih aksioma (detaljnije objašnjeno u poglavlju 4.). Model učenika je moguće predstaviti putem aksioma tipa BAKS.

Pored modela učenika, baza podataka ovog tipa sadrži skup pitanja i skup odgovora na ta pitanja. Pitanja i odgovori mogu biti pohranjeni u obliku tabela a to znači da se za implementaciju ovog dela baze podataka može upotrebiti neki od relacionih softvera za rukovanje bazama podataka (Access, Visual FoxPro, Visual Basic sa komponentom pristupa bazama podataka, ORACLE).

Obrazovni računarski softver za vežbanje

Karakteristike: ovaj tip softvera se koristi isključivo za utvrđivanje znanja pri čemu se korisniku ne saopštavaju nova znanja. Korisnik može i sam da vrši selekciju zadataka koje treba da reši.

U bazi podataka ovog tipa softvera sem skupa pitanja i odgovora iz odgovarajuće predmetne oblasti potrebno je da postoji model učenika. Za realizaciju ovog tipa softvera predlaže se relacioni model podataka odnosno softveri za rukovanje bazom zasnovani na tom modelu.

U [ĆNR96] je opisan postupak projektovanja testiranja znanja učenika korišćenjem tipa obrazovnog softvera za vežbanje. Obrazovni softver (konkretno - programski paket KSN4) realizovan je u relacionom softveru Visual FoxPro ver.3.0. Baza podataka sadrži relacije: PITANJA, ODGOVORI, UČENIK, OPSEG, VREME.

U [MK96] opisan je način kreiranja multimedijanog softvera korišćenjem generatora aplikacija, koji je projektovan na Učiteljskom fakultetu u Beogradu. Generator multimedijanog elektronskog udžbenika kreiran je u Visual Basic-u i omogućava nastavniku da definiše strukturu udžbenika; za svaku nastavnu jedinicu moguće je definisati tekst, tri slike (BMP fajlovi), video - klip (film - AVI fajl) i zvuk (WAV fajl). Baza podataka ovog softvera realizovana je korišćenjem softvera za baze podataka Access i sadrži sledeće tabele:

- Poglavlja,
- Lekcije,
- Test i
- Rezultati.

Obrazovni softver tipa banke podataka (znanja)

Karakteristike: ovaj tip softvera je, u stvari, specijalizovana enciklopedija znanja. Korisnik može i sam da vrši izbor prezentacije nastavnih sadržaja u skladu sa svojim potrebama.

Baza podataka ovog tipa softvera sadrži samo gradivo - model učenika nije potreban. Zbog potrebe da se u bazi podataka predstavi nastavno gradivo ne samo u obliku standardnih tipova podataka (znakovni, numerički, datumski) nego i u obliku slike, zvuka i animacija, za projektovanje ovog tipa softvera koristi se neki od relacionih softvera za rukovanje bazom ali koji ima mogućnost predstavljanja nestandardnih tipova podataka. Softveri kao što su Access, Visual FoxPro imaju mogućnost definisanja tipa obeležja koje može da predstavi sliku, zvuk i video zapis, kao što je napred objašnjeno.

Međutim, za realizaciju ovog tipa softvera moguće je koristiti ***objektno-orijentisani model podataka***. Ovaj model podataka je nastao delom i zbog potrebe memorisanja multimedijalnih zapisa: slika, grafika, digitalizovanih zvučnih i video zapisa. Ovi multimedijalni zapisi se u objektno-orijentisanim bazama podataka memorišu kao nizovi bajtova promenljive dužine. Delovi tih nizova su međusobno spregnuti, radi lakšeg povezivanja tekstova, slika, zvuka i video zapisa.
U [IYI96] dat je prikaz jednog objektno-orijentisanog sistema za baze podataka - **JASMIN**. Ovaj sistem je razvijen za potrebe aplikacija u CAD sistemima u kojima se podaci nalaze u obliku teksta, slika, tabela. U ovom softveru za rukovanje bazom podataka su ovi tipovi podataka definisani kao objekti. Podaci velike dužine su memorisani kao atributi tipa string veličine jednog bajta. Korisnik može da optimalno prilagodi maksimalnu dužinu podataka, koji se nalaze u jednoj stranici memorije, putem specifikacije veličine stranice za svaki objekat.

U ovom softveru se prevođenje objekata u relacije automatski izvodi od strane sistema. Instance klasa su memorisane u obliku relacija, tako da instanca odgovara torki a atribut polju u torki. Svaka instanca klase (torka) ima svoj OID (identifikator objekta) koji je predstavljen putem atributa koji, takođe, odgovara jednom polju torke. OID atribut se sastoji od identifikacionog broja baze podataka, id_broja klase i id_broja instance. Karakteristika ovog softvera je da je OID definisan logički i nema fizičku adresu u sekundarnoj memoriji (kao na primer u objektno-orijentisanom softveru za rukovanje bazama - O2). **Operacije** nad ovim tipovima podataka su definisane kao **metode objekata**. Heterogenim tipovima podataka je moguće manipulisati na jedinstven način putem **polimorfizma.**

Obrazovni softver tipa eksperimenta

Karakteristike: ovaj tip softvera se koristi za oglede u laboratoriji i u praktičnoj nastavi za merenje i upravljanje procesima, aparatima i mašinama.

Ovaj tip softvera ne zahteva postojanje modela učenika u bazi podataka. Realizacija ovog tipa softvera moguća je korišćenjem softverskih alata u matematici i računarskom projektovanju: Eureka, MathCAD, AutoCad ili je moguće razviti sopstvenu aplikaciju upotrebom nekog od strukturnih programskih jezika: C, Fortran, Pascal. U ovom tipu softvera se ne nameće zahtev za postojanjem baze podataka sa nekim specifičnim, određenim podacima.

Obrazovni računarski softver tipa simulacije

Karakteristike: putem ovog tipa softvera vrši se predstavljanje nekog realnog sistema na računaru putem adekvatno odabranog modela i vrši se simulacija procesa u tom sistemu i procesa iz njegovog okruženja. Takođe se predstavlja i struktura datog sistema. Korisnik može i sam da kreira modele za simulaciju sistema.

Ovaj tip softvera ne zahteva postojanje modela učenika.

Zbog potrebe za samostalnim kreiranjem modela datog realnog sistema od strane učenika, za realizaciju ovog tipa softvera podesan je objektno-orijentisani model podataka. Ovaj model podataka je odabran ne samo zbog mogućnosti predstavljanja heterogenih tipova podataka (kao što je to bio slučaj u obrazovnom softveru tipa banke podataka) nego i zbog toga što on poseduje sledeće mogućnosti [ML96]:

- upravljanje verzijama. Ova mogućnost je veoma važna u inženjerskom projektovanju gde tokom rada na jednom projektu nastaje više verzija rešenja problema.
- evolucija šeme. U inženjerskom projektovanju se veoma intenzivno menja statička struktura projekta što zahteva dinamičke izmene šeme baze podataka. Relacioni softveri za rukovanje bazama podataka imaju relativno skromne mogućnosti dinamičke izmene šeme baze podataka.
- ekvivalentni objekti. "Postoji više mogućih pogleda na isti objekat projektovanja. Na primer, jedan VLSI čip se može predstaviti na nivou logičkih kola, u cilju provere logičke funkcionalnosti, na nivou tranzistora, u cilju analize brzine rada, ili na nivou šeme povezivanja, u cilju provere primenjenih pravila projektovanja...Te reprezentacije se nazivaju *ekvivalentnim objektima*...Naime, ako se deo projekta izmeni u jednoj reprezentaciji, sistem treba ili da tu promenu sprovede i u drugim reprezentacijama, li da barem isti deo, u drugim rezprezentacijama označi kao nevažeći." ([ML96], str. 219) Ova osobina objektno-orijentisanog modela podataka je važna za simulaciju realnog sistema u obrazovnom softveru.

Posebnu ulogu u ovom tipu softvera ima savremena tehnologija - virtuelna realnost, opisana u [Nad97a]. Upotrebom tehnologije virtuelne realnosti kreiran je trodimenzionalni kompjuterski model Leonardovog ornitoptera. Ovaj pristup može biti osnova za istraživanja primene sličnih tipova simulacije i u obrazovanju.

Softverski alati

Karakteristike: ovaj tip softvera je predstavljen u obliku specijalizovanih programa kao što su: alati za obradu teksta (Word), formiranje baza podataka (Access, Visual FoxPro, Clarion, dBase, Visual Basic), tabelarni proračuni (Excel, QuattroPro, Lotus), grafički softverski paketi (Corel Draw, AutoCAD).

Rad korisnika se svodi na upotrebu konkretnog alata za određene, jasno definisane potrebe (kreiranje teksta, formiranje i ažuriranje sopstvene baze podataka).

Ekspertni sistemi

Karakteristike: Ovaj tip obrazovnog softvera predstavlja najviši nivo obrazovnog softvera i temelji se na ostvarenjima veštačke inteligencije. "Sistemi bazirani na znanju u oblasti obrazovanja imaju cilj da unaprede podučavanje, obučavanje, savetovanje i učenje korišćenjem metoda tutorijalne podrške samostalnom iznalaženju rešenja, didaktičkog objašnjavanja, upozoravanjem na greške, itd." [Vra94]

U [Hot95] su definisane glavne komponente ekspertnog sistema i to su:
1. Baza znanja
2. Radna baza (baza podataka)
3. Mehanizam zaključivanja
4. Veza sa korisnikom.

"Baza znanja sadrži na određen način strukturirano znanje iz poblemske oblasti. Najčešće je to deklarativno prezentirano znanje u obliku pravila implikativnog tipa ako...onda, frejmovske strukture ili semantičke mreže." ([Hot95], str. 197)

U [Dev94] baza znanja ekspertnog sistema je definisana kao logička celina, koja je sastavljena od tri međusobno povezana dela:
- znanje iz domena primene ekspertnog sistema (ili domenskog znanja),
- model ekspertnog sistema i
- deskriptor baze znanja.

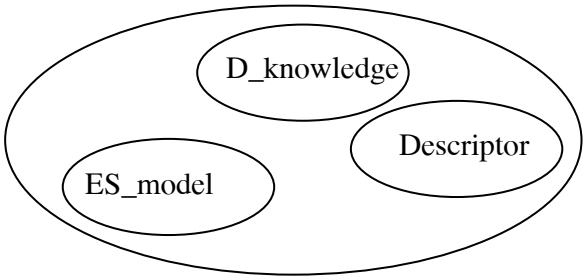
Domensko znanje je strukturirani zapis međusobno povezanih elemenata koji mogu da sadrže sve počev od jednostavnih podataka do kompleksnih elemenata znanja poput pravila, okvira, itd. Elementi domenskog znanja obično su grupisani u smislene kolekcije. Pojam "kolekcije", ovaj autor koristi umesto izraza poput "skupovi, "liste, "tabele", itd. da bi se sprečilo sugerisanje neke određene tehnike implementacije u fazi analize problema.

Baze podataka ekspertnih sistema su do sada, uglavnom bile realizovane upotrebom relacionog modela podataka. To znači da je za projektovanje i razvoj baze podataka ekspertnog sistema korišćen neki od konkretnih softvera za rukovanje bazama podataka: ORACLE, INGRES, Access.

U [Vra94] je opisan BEST - alat za izgradnju kompleksnih softverskih sistema baziranih na znanju. Baza podataka ovog alata je razvijena u relacionom softveru za baze podataka - Access-u, pa je moguće putem jednog specifičnog interfejsa (DDE - Dynamic Data Exchange) povezati ovaj sistem sa svim aplikacijama koje rade u istom okruženju (baze podataka, poslovni paketi, grafički paketi). Takođe, zahvaljujući standardnom protokolu (ODBC - Open Data Base Connectivity) moguće je pristupati bazama podataka kreiranim u drugim softverima za rukovanje bazama podataka, koje podržavaju taj protokol (ORACLE, SQL formati).

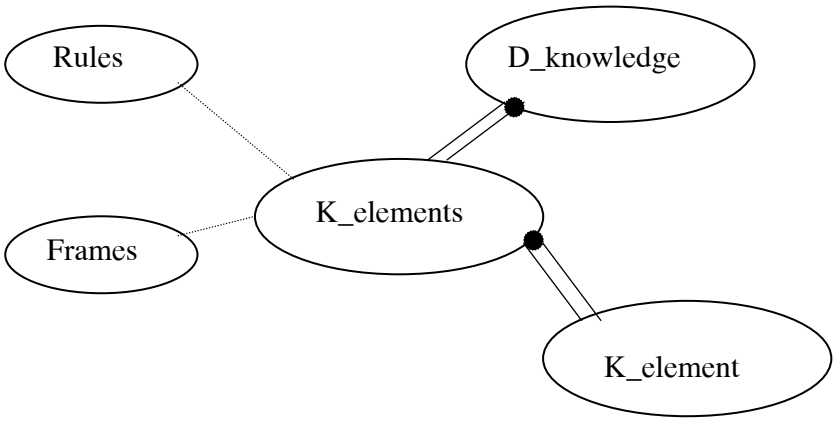
Međutim, baze znanja u ekspertnim sistemima zahtevaju predstavljanje podataka sa bogatom semantikom. Zbog toga je relacioni model podataka u izradnji baze znanja ES potrebno zameniti objektno - orijentisanim modelom podataka. U relacionom modelu podataka se semantika iz realnog sveta uglavnom predstavlja putem aplikativnih programa. Kada je potrebno dodati neko novo pravilo u bazu, sistem mora uvek da proverava da li ono već postoji ili je u suprotnosti sa postojećim pravilima.

Jedan pristup u definisanju objektno-orijentisanih baza znanja dat je u [Dev94]. U ovom radu se baza znanja ekspertnog sistema definiše kao ***kompozitni objekat*** na način, ilustrovan na slici 2.1.:



Slika 2.1.

Sledeći dijagram pokazuje kako se elementi baze znanja organizuju i grupišu u homogene kolekcije K_elements, prema [Dev94]. Objekat iz klase K_elements je kolekcija objekata iz klase K_element. Klasa K_element opisuje pojedinačne elemente domenskog znanja, što je predstavljeno na slici 2.2.



Slika 2.2.

Dvostruka linija od klase D_knowledge do klase K_elements sa tačkom uz D_knowledge označava da klasa D_knowledge koristi klasu K_elements.

Elementi klase D_knowledge imaju sledeće svojstvo: jedan objekat iz klase D_knowledge dozvoljava drugim objektima (koji komuniciraju sa njim) da koriste funkcije iz njegovog interfejsa da bi definisali ili pročitali njegovo ime, očitali njegovu veličinu, pristupali kolekcijama njegovih elemenata znanja itd. Takođe, u ovom pristupu projektovanju baza znanja ES vodi se računa o enkapsulaciji (učaurivanju). To je generalni i važan princip objektno-orijentisanog projektovanja softvera: "najvećem broju atributa (polja) klasa i objekata u ovakvoj organizaciji baze znanja ne može se pristupati direktno. Objekti-klijenti moraju da koriste funkcije interfejsa objekta-servera".

Obrazovni softver tipa virtuelne realnosti

Karakteristike: U [Nad97a] virtuelna realnost je definisana kao: "Virtuelna realnost je trodimenzionalno, kompjuterski generisano, simulirano okruženje koje je postavljeno u realnom vremenu u skladu sa ponašanjem korisnika...Virtuelna realnost biće deo svega što radimo, uključujući učenje." Takođe su objašnjeni alati i tehnike putem kojih se realizuje virtuelna realnost: hardverske platforme, softverski alati i aplikacije, objektno-orijentisani dizajn multimedije.

Definisane su karakteristike virtuelne realnosti:
- trodimenzionalnost,
- kompjuterski generisanost,
- simulirano okruženje,
- smeštena je u realnom vremenu u skladu sa ponašanjem korisnika.

Tehnologija virtuelne realnosti ima ulogu i u kreiranju obrazovnog softvera tipa simulacije (kao što je napred rečeno) i tipa obrazovne igre (u daljem tekstu).

2.2.2. Mogućnosti primene baza podataka za realizaciju ciljeva i zadataka obrazovnih softvera u odnosu na način korišćenja kompjutera u obrazovanju

Prema klasifikaciji koju je dao Tejlor u [Te89], u osnovi postoje tri načina korišćenja kompjutera u obrazovanju:
1. kompjuter kao učitelj,
2. kompjuter kao korisničko sredstvo za rad i
3. kompjuter kao sredstvo koje korisnik uči da bi on učio sebe i druge.

2.2.2.1. Obrazovni softver za kompjuter koji ima ulogu učitelja

U ovom tipu obrazovnog softvera prisutan je sledeći scenario: izlaže se određeni obrazovni sadržaj učeniku, učenik odgovara, softver ocenjuje odgovor i na osnovu rezultata odgovora softver određuje dalje faze rada učenika. Kod boljih rešenja tutorskih programa predviđeno je čuvanje podataka o podučavanom učeniku. U ovoj grupi softvera postoje sledeće vrste:
- Dril i vežbanje,
- Tutorski programi,
- Simulacija i stvaranje modela,
- Rešavanje problema,
- Obrazovne igre.

Dril i vežbanje

Karakteristike: ovaj tip softvera se koristi za utvrđivanje ranije obrađenog obrazovnog sadržaja i ne prezentuje se novi. Postavljaju se pitanja učeniku, ocenjuje se njegov odgovor, obezbeđuje se ograničena povratna sprega i zatim se postavlja novo pitanje uzimajući u obzir dati odgovor.

S obzirom da u ovom tipu softvera postoji određena povratna sprega potrebno je da u bazi podataka postoji model učenika. Za implementaciju baze podataka ovog tipa softvera predlaže se relacioni model podataka.

Primer 2.2. Model učenika je predstavljen u vidu sledećih šema relacija:

UČENIK_OPŠTIPOD({MBR_UČENIKA,IME,PREZIME,RAZRED,GOD_ROD},{MBR_UČENIKA})
UČENIK_ODGOVORI({MBR_UČENIKA,BR_PITANJA,BR_BODOVA},
{MBR_UČENIKA, BR_PITANJA})

S obzirom da ovaj tip softvera postavlja pitanje učeniku uzimajući u obzir date odgovore na prethodna pitanja, u bazi podataka ovog tipa softvera potrebno je kreirati sledeće šeme relacije:

PITANJA({BR_PITANJA, OPIS_PITANJA, SLED_PITANJE},{BR_PITANJA})

Značenje obeležja *sled_pitanje* u relaciji PITANJA je: broj pitanja koje će softver postaviti u narednom koraku učeniku ako je učenik odgovorio tačno na dato pitanje. To znači, da se realizacija povratne sprege u ovom tipu softvera može realizovati procedurno. Ako je učenik odgovorio na dato pitanje i osvojio određeni broj bodova (ta informacija se nalazi u relaciji UČENIK_ODGOVORI) prelazi se na sledeće pitanje, koje je nastavnik odredio u zavisnosti od ciljeva časa i semantike nastavnog gradiva. □

Rešenje koje se ovde predlaže zasniva se na procedurnoj realizaciji provere osvojenog broja bodova od strane učenika za dato pitanje i na osnovu toga zadavanje novog pitanja učeniku. Projektant obrazovnog softvera i baze podataka ima obavezu da u saradnji sa nastavnikom definiše kojim se redosledom zadaju pitanja učeniku (određivanje vrednosti obeležja *sledeće_pitanje* u relaciji PITANJA). To znači da se od projektanta baze podataka i nastavnika zahteva dobro definisana ***sistem analiza*** nastavnog gradiva u vidu redosleda pitanja.

Međutim, umesto procedurne realizacije redosleda pitanja učenika u obrazovnom softveru tipa drila i vežbanja, moguć je i logičko-deklarativni prilaz. U razvoju baze podataka ovog tipa obrazovnog softvera moguće je iskoristiti sistem DEDUC - sistem za kombinatorno generisanje rasporeda [Hot97], [Hot92], [Pro94.]. "Logičko-deklarativni prilaz omogućuje da se procedurni deo sistema učini nezavisnim, a time i nepromenljivim, u odnosu na variranje zahtevanih uslova u konkretnom slučaju".

Realizacija ovog tipa softvera upotrebom sistema DEDUC zahteva od projektanta definisanje uslova pod kojima se generiše redosled zadavanja pitanja učeniku. Te uslove određuje nastavnik u zavisnosti od strukture nastavnog gradiva. Uslovi se nalaze u aksiomatskoj bazi, čiji su elementi predikatske formule (sastavci). Na taj način se može definisati težina pitanja koja će se postaviti učeniku u zavisnosti od njegovih karakteristika i sposobnosti. Karakteristike učenika (opšte i posebne) se nalaze u vidu vrednosti obeležja šeme relacije UČENIK u relacionoj bazi podataka (DBF format FoxPro softvera).

Primer 2.3. Predikat D(N,P,Q,d,t) sistema DEDUC u slučaju projektovanja ovog tipa softvera može da se zameni sa sledećim predikatom:

Sled_pitanje(U,P,O,sled_pit)

koji ima značenje: učeniku U, koji je na pitanje P odgovorio tačno (O) postavlja se sledeće pitanje *sled_pit*. Konkretni podaci se nalaze u bazi podataka u vidu šeme relacije:

TEST({ID_UČENIKA,ID_PITANJA,ODGOVOR},{ID_UČENIKA,ID_PITANJA})

To znači da su U, P i O konstante a *sled_pit* je promenljiva, koja se u procesu određivanja sledećeg pitanja zamenjuje konstantom.

Tutorski programi

Karakteristike: ovaj tip softvera se koristi za učenje novog gradiva i utvrđivanje prethodno savladanog. On ima opciju: provera učenikovih odgovora i davanje uputstava o daljem toku učenja. Poseban tip softvera čine ***inteligentni tutorski sistemi*** i ti sistemi imaju karakteristike ekspertnih sistema. U njima se na poseban način strukturira znanje koje učenik treba da savlada. Ovi sistemi, takođe, sadrže tipove grešaka, njihovo poreklo
i sugestije u vezi sa greškama. Na bazi grešaka koje prave učenici usmerava se dalji tok nastavnog procesa. Sam softver odlučuje o tome kakvu vrstu pomoći treba dati učeniku, koje zadatke treba postaviti u narednim koracima učenja.

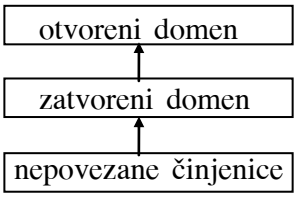
U bazi podataka ovog tipa softvera postoji model učenika. U [JD96] studentov model inteligentnog tutorskog sistema EduSof konceptualno je zamišljen kao tabela koja sadrži ubeđenje studenta (učenika) da poseduje neko znanje iz domena. Ova tabela je kopija domenskog stabla sa tom razlikom da je svaki čvor u stablu studentovog modela - verovanje, ubeđenje studenta da on zna znanje predstavljeno tim čvorom.

U didaktičkoj teoriji definisano je još nekoliko modela učenika, kao komponente inteligentnih tutorskih sistema i to su: model preklapanja (overlay model), model grešaka (buggy model), model objašnjavanja, rekonstruktivni model. Razlike između ovih modela se ogledaju u načinu putem koga se predstavlja znanje učenika u odnosu na znanje eksperta:

- kao podskup (model preklapanja),
- kao skup znanja koje ima učenik ali ih nema odgovarajući ekspert (model grešaka)
- kao razlika učenikovog znanja u odnosu na znanje eksperta (rekonstruktivni model).

Projektovanje baze podataka ovog tipa softvera moguće je upotrebom logičkog modela podataka. BASELOG-sistem, opisan u poglavlju 4. je adekvatan za praktičnu implementaciju modela učenika. U poglavlju 5. opisan je postupak na osnovu koga se formiraju aksiome tipa BAKS na osnovu baze podataka, koja je kreirana upotrebom *relacionog softvera za rukovanje bazama podataka* (DBF format). U ovoj bazi podataka nalaze se osnovne informacije o učenicima (opšti podaci, odgovori učenika, reakcije učenika, stepen prethodnog znanja, tekuće znanje). Pravila zaključivanja inteligentnog tutorskog sistema moguće je predstaviti putem sopstvenih aksioma BASELOG-sistema.

Važno je napomenuti da se u didaktičkoj literaturi [El84] sreće i pojam *otvorenog* i *zatvorenog domena*. Naime, u tom radu je problemski domen tutorskih sistema predstavljen na sledeći način:



Najjednostavniji domen su **nepovezane činjenice**. Tutorski sistemi, koji imaju ovaj tip problemskog domena, ne mogu obezbediti učenicima informacije o problemima, koji se nalaze van znanja u sistemu. Prethodno znanje učenika nema efekta na proces učenja.

Kompleksniji domen je **"zatvoreno okruženje"** u kome se nalaze sve moguće adekvatne metode povezanih činjenica. Obezbeđivanjem redukovanog skupa mogućih znanja, ovakvo okruženje u tutorskom sistemu pojednostavljuje problem projektovanja modela učenika i omogućava da se sistem ponaša kao da ima kompletno znanje o datom problemu. Prethodno znanje učenika ima efekta na proces učenja. Ovakav tip domena dopušta projektovanje "eksperta" i omogućava generisanje skupa mogućih "eksperata" domena. Većina tutorskih sistema rade u "zatvorenom" okruženju.

"Otvoreni domen" je onaj tip domena u kome nije ograničen broj mogućih problema. Obrazovni tutorski softver koji radi u ovom okruženju nema kompletno znanje ni sve metode za rešavanje problema iz date oblasti. Ove karakteristike utiču na projektovanje modela učenika i procenu učenikovog napredovanja. Iz ovih razloga se u projektovanju tutorskih sistema zahteva da učenici koriste takve mehanizme u procesu svog učenja, koje su tutorskom sistemu poznate, čak i kad ti mehanizmi nisu adekvatni ni za učenike ni za dati domen.

U [El84] se, takođe, navodi da zbog ograničenja na odgovarajući tip domena u kome sistem radi, mora se voditi računa i o tome kako će se učenici snalaziti u tom domenu. Izbor tipa domena mora biti takav da je moguće videti kako će on uticati na opšte mogućnosti projektovanja sistema. Takođe je stalno prisutan zahtev za postojanjem jasne razlike između otvorenog i zatvorenog tipa domena.

Opšti stav u didaktici je da nije moguće predstaviti sve probleme i rešenja u domenu, tako da individualno domensko znanje mora postojati u obliku nekog opšteg modela domena, koji oponaša ponašanje u raznim uslovima. Nastavnik mora biti spreman da prihvati bilo koji model domena. Domensko znanje se najčešće predstavlja u obliku semantičkih mreža, hijerarhijskih struktura - generalizacija, frejmova.

Znanje domena je statično, iako postoje generalne veze među komponentama, koje modelujemo. U [El84] se navode sledeći primeri predstavljanja znanja u tutorskim sistemima:

- SCHOLAR, WHY i TRILL koriste semantičke mreže za predstavljanje koncepata znanja. Teme (iz nastavnog gradiva) su predstavljene putem hijerarhijskih struktura, koje u svojoj osnovi imaju nivoe generalizacije. Pojedinačna tema može biti podkomponenta opšteg (generalnog) koncepta i može da sadrži druge teme kao svoje podkomponente.
- Genetic Graph ima domensko znanje koje je reprezentovano u vidu malih "ostrva" znanja. Ove jedinice znanja su povezane vezama koje odgovaraju različitim metodama učenja, kao što su generalizacija, analogija itd.
- Tutorski sistem GUIDON koristi ekspertni sistem MYCIN kao izvor domenskog znanja.

Savremeni pravci u razvoju tutorskih sistema idu ka tome da se ekspertni sistem proširi u nameri da uključi znanje koje treba da obezbedi podršku tutorskim instrukcijama.

U [JD96] opisan je inteligentni tutorski sistem EduSof. Domensko znanje ovog sistema je predstavljeno u obliku stabla, čiji svaki čvor predstavlja deo domenskog znanja i svaka
veza roditelj-dete znači da ako je student u mogućnosti da razume znanje, predstavljeno čvorom-roditelj student, takođe, mora razumeti i znanja koja su predstavljena putem čvorova-sledbenika (dece). Domensko znanje je predstavljeno putem objektno-orijentisane metodologije kao što je to objašnjeno u odeljku o inteligentnim sistemima.

Međutim, projektovanjem baze podataka obrazovnog softvera u kome se koristi BASELOG-sistem moguće je raditi i u okruženju otvorenog, zatvorenog i poluotvorenog domena (sveta). Koncepti projektovanja baze podataka obrazovnog softvera u režimima otvorenog, zatvorenog ili delimično otvorenog sveta objašnjeni su u poglavlju 4. dok je u poglavlju 5. naveden praktičan primer projektovanja baze podataka upotrebom BASELOG-sistema.

Simulacija i stvaranje modela

Ovaj tip softvera je objašnjen u klasifikaciji softvera po Đ. Nadrljanskom. Potrebno je samo istaći da postoji razlika između simulacije, kod koje korisnik treba sam da odredi model i simulacije, kod koje korisnik radi sa zadatim modelom. Ovaj tip softvera se koristi u prirodnim naukama (biologija, fizika, hemija) i u društvenim naukama (ekonomija). Kao što je rečeno, za projektovanje baze podataka ovog tipa softvera adekvatan je objektno - orijentisani model podataka (rukovanje heterogenim podacima, upravljanje verzijama, evolucija šeme, ekvivalentni objekti). Za realizaciju simulacije u ekonomiji mogu da se koriste i gotovi softverski paketi za tabelarne proračune (Excel, QuattroPro, Lotus).

Rešavanje problema

Karakteristike: učenje upotrebom ovog tipa softvera se realizuje putem kreiranja algoritma za rešenje problema koga softver postavlja korisniku. Ponekad se zahteva i poznavanje principa programiranja i programskog jezika. U slučaju složenijih problema, od korisnika se traži i analiza grešaka u strategiji rešavanja problema. Ovaj tip softvera ne zahteva postojanje modela učenika. Njegova realizacija moguća je upotrebom nekog od programskih jezika (C, Fortran, Pascal) ili se mogu koristiti specijalizovani paketi za inženjerske i matematičke proračune (MathCad, Eureka).

Obrazovne igre

Karakteristike: ovaj tip softvera razvija veštine manipulisanja podacima, istraživanja, planiranja, analiziranja, postavljanja hipoteza, otkrivanja, posmatranja. Posebna vrsta su avanturističke igre putem kojih se razvija osećaj za prostor, sposobnost donošenja odluka, rešavanja problema.

Ovaj tip softvera se može realizovati sa ili bez modela učenika i njegova implementacija je moguća upotrebom grafičkih paketa sa mogućnostima višedimenzionalnih animacija i alata za reprodukciju zvuka. Posebnu ulogu imaju sistemi virtuelne realnosti, opisani u [Nad97a]. Za potrebe razvoja ovog tipa softvera koriste se softveri za rukovanje bazama podataka sa mogućnošću predstavljanja nestandardnih tipova podataka (slika, zvuk, animacije). Objektno-orijentisani softveri za rukovanje bazama podataka, kao što je napomenuto prilikom opisa obrazovnog softvera tipa simulacije, imaju mogućnost memorisanja ovakvih tipova podataka.

2.2.2.2. Obrazovni softver za kompjuter kao sredstvo za rad

U ovoj grupi softvera se nalaze softveri za statističku analizu, obradu teksta, grafički paketi, softveri za rukovanje bazama podataka.

Pretraživanje podataka

Karakteristike: u obrazovnim sadržajima nekih predmeta (uglavnom predmeta iz oblasti društvenih nauka), u kojima se radi sa većim obimom podataka, učenici mogu da koriste softvere za rukovanje bazama podataka u cilju kreiranja sopstvene baze podataka ili da samo pretražuju postojeću bazu podataka. Softveri za baze podataka (dBase, FoxPro) su veoma pogodni za početnike, jer imaju dobro razvijen korisnički interfejs i vodič tutorskog tipa, koji korisnika vodi kroz sve faze razvoja aplikacije za kreiranje i ažuriranje baza podataka.

Kompjuterski vođeno učenje

Karakteristike: kompjuter se koristi u učionici za upravljanje pomoćnim nastavnim sredstvima u tradicionalnoj nastavi, ili za kreiranje testa, ili za praćenje napredovanja učenika.

U ovom tipu softvera postoji model učenika i njegova realizacija je moguća putem relacionog modela podataka. U bazi podataka ovog tipa softvera moguće je kreirati relacije za memorisanje opštih podataka o učeniku (ime, prezime, godina rođenja, razred) i posebnih podataka (za datu nastavnu jedinicu (temu) broj bodova osvojenih na testu, broj bodova za prethodno urađene testove, ocena, predlozi nastavnika, predlozi učenika).

Kompjuter kao instrument ili laboratorija
O ovom tipu softvera je bilo reči u klasifikaciji po autoru Đ. Nadrljanskom.

2.2.2.3. Softver za kompjuter kao sredstvo koje pomaže učeniku da uči i da bi učenik učio druge

Ovaj tip softvera je kreiran od strane naprednijih učenika sa namenom da taj softver primenjuju početnici. Kreiranjem ovog softvera povećavaju se sposobnosti učenika, koji kreiraju softver, sa nivoa usvajanja činjenica na nivo razumevanja i manipulisanja činjenicama.

Za realizaciju ovog tipa softvera adekvatan je programski jezik LOGO. Učenici razvijaju sposobnost za rešavanje problema tako što se problem razloži na njegove sastavne komponente, a zatim se rešava svaka od njih posebnom procedurom.

Istraživanje i otkriće

Karakteristike: učenici upotrebom softvera ovog tipa istražuju i otkrivaju zakonitosti ili relacije nad podacima koji se nalaze u bazi podataka obrazovnog softvera.

Primer 2.4. Softver raspolaže podacima o selima u kojima se kontroliše širenje bolesti. Učenik preuzima ulogu medicinskog stručnjaka u jednom selu i uz pomoć programa pronalazi rešenja za sprečavanje epidemije.

Realizacija baze podataka ovog tipa softvera moguća je upotrebom relacionih softvera za rukovanje bazama podataka (Access).

2.3. Modeli podataka u realizaciji pojedinih elemenata koji mogu biti prisutni u raznim tipovima ORS

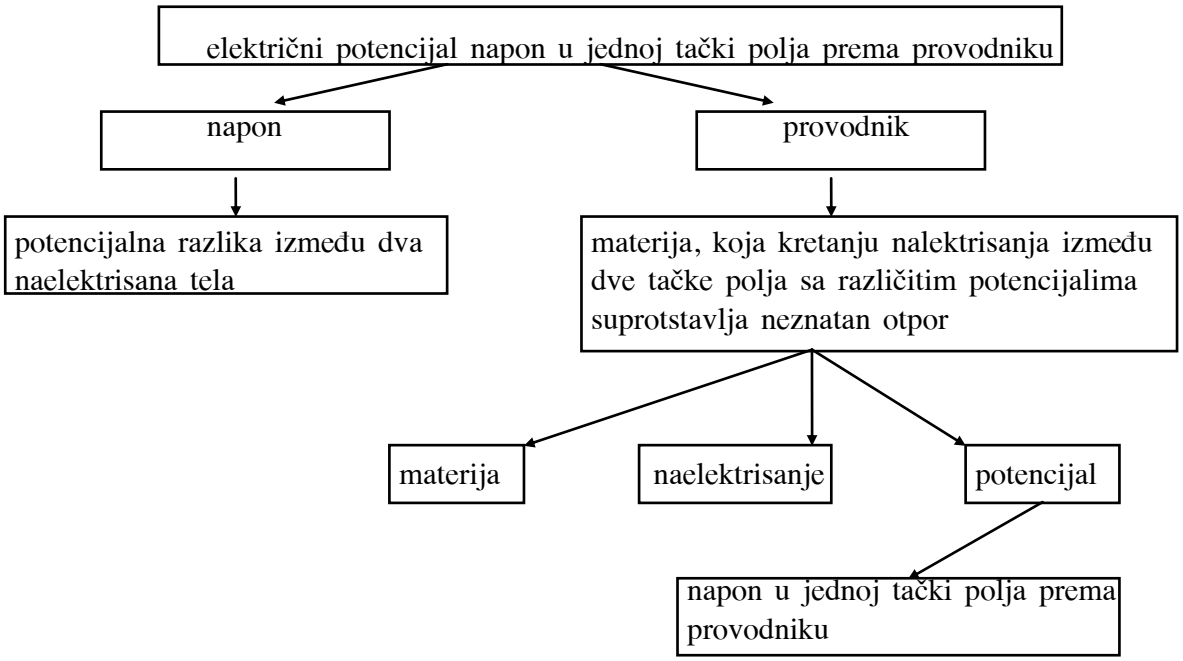
U ovom odeljku su opisani karakteristični postupci projektovanja baza podataka za potrebe obrazovnog softvera putem sledećih modela podataka: relacionog, objektno-orijentisanog, logičkog i kombinacijom relacionog i logičkog modela. Izbor modela podataka je izvršen u odnosu na semantiku baza podataka i ciljeve koje treba da ostvari obrazovni softver. Opisani su sledeći postupci:

- **Dekompozicija pojmova.** Opisan je postupak projektovanja na osnovu relacionog modela podataka u slučaju da se radi o dekompoziciji *pojmova*. Takođe, opisan je postupak projektovanja i putem objektno-orijentisanog modela podataka.
- **Provera učeničkih hipoteza u dijaloškim sistemima.** Opisan je postupak projektovanja baza podataka primenom logičkog modela podataka.
- **Kreiranje (projektovanje) modela korisnika.** U ovom postupku je izvršena kombinacija relacionog i logičkog modela. Naime, tip softvera, koji u sebi ima model korisnika treba da omogući izvođenje zaključaka u odnosu na korisnika: koje informacije treba dati korisniku u narednim fazama učenja, ili koje sledeće pitanje postaviti u narednom koraku testiranja (vežbanja).

2.3.1. Projektovanje baza podataka za slučaj dekompozicije pojmova

Dekompozicija pojma je postupak razlaganja definicije nekog pojma na jedan ili više pojmova koji učestvuju u definiciji. Time se dobija struktura tipa stabla u kojoj su elementi listova stabla fundamentalne (elementarne) definicije. Fundamentalne definicije su one, čijom bi se daljom dekompozicijom izgubio smisao pojmova, posmatrano u odnosu na definiciju pojma na najvišem nivou hijerarhije (koji se nalazi u korenu ovakvog stabla). Namera ovog postupka je da se učeniku omogući izbor definicija onih pojmova za koje on smatra da su mu neophodne u toku usvajanja novog pojma. Učenik sam bira put kojim usvaja nove pojmove. Takođe, predstavljanjem nastavnog gradiva u obliku hijerarhijske strukture dobija se na preglednosti i lakšem postupku projektovanja nastavne baze podataka. To je ilustrovano sledećim primerom.

Primer 2.5. Prema [Il78] pojam ***električni potencijal*** je definisan kao napon u jednoj tački polja prema provodniku. Ovaj pojam se može dekomponovati na sledeći način u obliku hijerarhijske strukture:



U didaktici se nastavno gradivo predstavlja u obliku sledeće hijerarhijske strukture:

- nastavno područje,
- nastavna oblast,
- nastavna tema,
- nastavna jedinica i
- didaktička celina,

pri čemu je nastavno područje koncept na najvišem nivou složenosti a didaktička celina koncept na najnižem nivou.

Polazeći od ovakve hijerarhijske strukture zaključuje se da se u bazi podataka neke nastavne oblasti nalaze sve definicije (elementarne i složene), identifikacija nastavnih jedinica (naziv) i definisana pripadnost pojmova nastavnim jedinicama.

Primer 2.6. Uzeto je *nastavno područje* fizike za VIII razred osnovne škole [Il78]. U skladu sa nastavnim planom za ovaj razred definisane su sledeće nastavne oblasti:

- elektromagnetne pojave,
- svetlosne pojave,
- nuklearni procesi.

U *nastavnoj oblasti* **Elektromagnetne pojave** izdvajaju se sledeće nastavne teme:

- električno polje,
- električna struja,
- magnetno polje,
- elektromagnetno polje i njegove primene.

U *nastavnoj temi* **Električno polje** moguće je identifikovati sledeće nastavne jedinice:

- uzajamno dejstvo naelektrisanih tela,
- objašnjenje naelektrisanja,
- električno polje,
- provodnici, izolatori, poluprovodnici,
- provodnik u električnom polju i električna influencija,
- raspored naelektrisanja na provodnicima,
- električni kapacitet provodnika; kondenzatori.

U *nastavnoj jedinici* **Električni kapacitet provodnika.Kondenzatori** izdvajaju se sledeće didaktičke celine:

1) Električni kapacitet provodnika.

Električni kapacitet provodnika brojno je jednak količini naelektrisanja, kojom treba naelektrisati taj provodnik da bi mu se potencijal povisio za 1 volt.

2) Formule

$$C = q / \varphi \qquad 1F = C / V$$

3) Kapacitet

Kapacitet od 1 farada ima provodnik čiji se potencijal promeni za 1 volt kada mu se dovede količina elektriciteta od 1 kulona.

4) Kondenzator

Kondenzator elektriciteta sastoji se od dva metalna provodnika između kojih se nalazi izolator.

Baza podataka za nastavne oblasti jednog nastavnog područja sastoji se iz sledećih šema relacija:

POJMOVI({SIF_POJMA,TEKST,SLIKA,ANIMACIJA, ZVUK},{SIF_POJMA})

Pojava nad ovom šemom relacije su definicije svih pojmova date nastavne oblasti.

NASTAVNE_JEDINICE({SIF_NASJED,NAZIV},{SIF_NASJED})

Pojava nad ovom šemom relacije je relacija u kojoj se nalaze imena svih nastavnih jedinica za datu nastavnu oblast.

POJMOVI_NASJED({SIF_NASJED,SIF_POJMA},{SIF_NASJED,SIF_POJMA})

Referencijalni integriteti su definisani u daljem tekstu.

Pojava nad ovom šemom relacije je relacija sa sledećom semantikom:

U nastavnoj jedinici, identifikovanoj putem obeležja *sif_nasjed*, relevantni su pojmovi, identifikovani putem obeležja *sif_pojma*. Ključ ove šeme relacije je složen i sastoji se od obeležja *sif_nasjed* i *sif_pojma*. To je posledica semantike po kojoj se jedna nastavna

jedinica sastoji iz više pojmova i svaki pojam može biti relevantan za više nastavnih jedinica.

DEKOMPOZICIJA({SIF_POJMA1,SIF_POJMA2}, {SIF_POJMA1,SIF_POJMA2})
Semantika ove šeme relacije je, takođe, posledica organizacije nastavnog gradiva, po kojoj jedan pojam može biti dekomponovan na više podpojmova i svaki pojam može biti deo definicije nekog drugog pojma.

Za slučaj *nastavne jedinice* **Električni kapacitet provodnika.Kondenzatori** pojave nad ovim šemama relacija imaju sledeći izgled:
relacija POJMOVI

SIF_POJMA	TEKST	SLIKA	ANIMACIJA	ZVUK
01	Električni kapacitet provodnika brojno je jednak količini naelektrisanja....	bmp fajl	avi fajl	wav fajl
02	Kapacitet od 1 farada ima provodnik čiji se...			
03	Naelektrisanje je količina elektriciteta koja proteče kroz...			
04	Provodnik je materija, koja kretanju naelektrisanja između...			
05	Potencijal u nekoj tački električ. polja brojno je jednak...			
06	Električno polje je prostor...			
07	Rad je rezultat delovanja sila i puta na kome ta sila deluje.			
08	Naelektrisano telo je telo koje posle trenja privlači druga tela.			
09	Elektron je najmanje negativno naelektrisanje.			
10	Sila - mera uzajamnog delovanja tela koje menja stanje kretanja.			
11	Volt je ...			
12	Farad je ...			
13	Kulon je ...			
14	Elektricitet je...			
15	q je ...			
16	φ je ...			
17	Kondenzator je...			
18	Izolator je ...			
.....	.....			

relacija NASTAVNE_JEDINICE

SIF_NASJED	NAZIV
A	Električni kapacitet provodnika. Kondenzatori.
B	Raspored naelektrisanja na provodnicima
C	Električno polje
.....	.....

relacija POJMOVI_NASJED

SIF_NASJED	SIF_POJMA
A	01
A	02
.....	.....

relacija DEKOMPOZICIJA

SIF_POJMA1	SIF_POJMA2
01	03
01	04
01	05
01	11
02	12
02	04
02	05
02	11
02	14
02	13
03	15
04	03
04	05
04	16
.....	.....

Pogodnost primene relacionog modela podataka za projektovanje baza podataka čija je semantika dekompozicija pojmova biće objašnjena putem sve tri komponente modela podataka.

1. strukturalna komponenta:

ekstenzija

- **torka** - jasna reprezentacija pojmova u vidu n-torki
  Na primer, POJMOVI(18,"Izolator je...",SLIKA,ANIMACIJA,ZVUK)
- **domen obeležja** - za slučaj primitivnih obeležja (tekstualni opis pojma) uglavnom je u pitanju tekst; za slučaj grafičkog predstavljanja pojma ili animacija radi se o nestandardnim tipovima podataka (obično su to nizovi bajtova promenljive dužine)
- **relacija** - putem relacija se predstavljaju didaktički koncepti na pregledan način
- **pojava baze podataka** - omogućava predstavljanje čitavog nastavnog područja u vidu skupa relacija

intenzija

- **obeležje** - definisanje elementarnih koncepata: nastavnih tema i jedinica
- **šema relacije** - omogućava definisanje i ograničenja: jedinstveno identifikovanje koncepata (nastavna tema, jedinica i dr.) putem ključeva za svaki koncept,
  Na primer, NASTAVNE_JEDINICE({SIF_NASJED,NAZIV},{SIF_NASJED})
- **šema baze podataka** - definisanje međurelacionih ograničenja što će se videti na primeru dekompozicije.

2. strukturalna ograničenja:

- **ograničenja torke** - ograničenje na dužinu obeležja za tekstualni opis pojma
- **ograničenja relacije** - jedinstveno identifikovanje nastavnih tema i drugih didaktičkih koncepata putem ključeva

Putem ova dva ograničenja definisana je *lokalna konzistentnost* baze podataka - a to znači da li pojava nad nekom šemom relacije zadovoljava ograničenja, koja su definisana u šemi relacije.

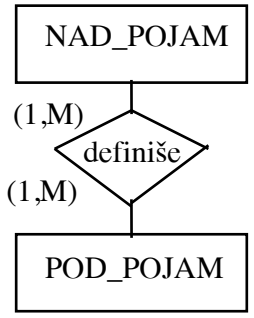
- **međurelaciona ograničenja**

Putem ovih ograničenja definiše se *globalna konzistentnost* baze podataka a to znači da li pojava nad konkretnom bazom podataka zadovoljava ograničenja koja su definisana u šemi baze podataka.

Primer 2.7, Kreirana je baza podataka za nastavnu oblast *Elektromagnetne pojave* iz **primera 2.6**. Polazeći od koncepta dekompozicije pojmova date nastavne oblasti u vidu skupa šema relacija, skup međurelacionih ograničenja ima sledeći izgled (notacija je preuzeta iz [ML96]):

$(\forall t \in r(Pojmovi\_NasJed))(\exists u \in r(Nastavne\_jedinice))\ (t[Sif\_NasJed]=u[Sif\_NasJed])$
$(\forall t \in r(Pojmovi\_NasJed))(\exists u \in r(Dekompozicija))\ (t[Sif\_pojma]=u[Sif\_pojma1])$
$(\forall t \in r(Dekompozicija))(\exists u \in r(Pojmovi))\ (t[Sif\_pojma1]=u[Sif\_pojma] \wedge$
$t[Sif\_pojma2]=u[Sif\_pojma])$

Dekompoziciju pojmova moguće je predstaviti putem dijagrama ER modela podataka na sledeći način kao na slici 2.3.:



Slika 2.3.

Kardinaliteti preslikavanja (1,M), koji se nalaze sa obe strane tipa poveznika ***definiše,*** označavaju situaciju u kojoj je svaki pojam moguće razložiti na jedan ili više pojmova i svaki pojam može učestvovati u definisanju jednog ili više pojmova. Saglasno pravilima za prevođenje ER modela podataka u relacioni model, ovaj dijagram se prevodi u sledeće šeme relacija:

Nad_pojam(Sif_Pojma1, *ostala_obeležja*)
Pod_pojam(Sif_Pojma2,*ostala_obeležja*)
Dekompozicija(Sif_Pojma1, Sif_Pojma2)

pri čemu je u primeru 2.6. definisana samo jedna relacija *Pojmovi* za predstavljanje svih pojmova u bazi podataka. To znači da ona obuhvata svojim sadržajem relacije Nad_pojam i Pod_pojam.

Softveri za rukovanje bazama podataka zasnovani na relacionom modelu podataka imaju mehanizme za kontrolu referencijalnih integriteta, što znači da se ne može uneti u relaciju *Dekompozicija* neka vrednost obeležja *Sif_pojma1* a da ona ne postoji u relaciji *Pojmovi_NasJed*, ili, ne može se uneti vrednost obeležja *Sif_pojma2* a da ona već ne postoji u relaciji *Pojmovi*. To znači da se automatski proverava od strane sistema konzistentnost ovako projektovane baze. Takođe, to je slučaj i kod brisanja i izmene torki u relacijama.

Na primer, u Visual FoxPro-u je to definisano u okruženju Database Designer. U ovom softveru su moguće sledeće opcije prilikom definisanja referencijalnih integriteta:

- kaskadno
- restriktivno
- ignorisanje.

U relacionom softveru za rukovanje bazama podataka - Access-u definisanjem veze među relacijama (Enforce Referential Integrity) obezbeđena je automatska kontrola unosa novih vrednosti u relaciju ako ne postoji odgovarajuća vrednost primarnog ključa u relaciji, sa kojom je data relacija u vezi. Selektovanjem opcija *Cascade Update (Delete) Related Records* moguće je definisati i uslove referencijalnog integriteta za slučaj izmene i brisanja torki u relacijama među kojima je uspostavljena veza, odnosno među kojima je definisan određeni tip poveznika, koji se predstavlja u Access-u konceptom *Relationship*.

Potrebno je naglasiti da su ovo specifičnosti pomenutih relacionih softvera za rukovanje bazama podataka. Pored nabrojanih opcija za održavanje referencijalnih integriteta, u relacionim sistemima za rukovanje bazama podataka važne su i sledeće opcije:

- NULLIFIES - ako se menja vrednost primarnog ključa neke torke ili se ta torka briše, a vrednost primarnog ključa te torke se nalazi kao strani ključ u drugim relacijama, moguće je vrednosti tih stranih ključeva ažurirati na nul-vrednost (pod uslovom da je dozvoljena nul-vrednost za ključeve tih relacija),
- DEFAULT - ako se menja vrednost primarnog ključa neke torke ili se ta torka briše, a vrednost primarnog ključa te torke se nalazi kao strani ključ u drugim relacijama, moguće je vrednosti tih stranih ključeva ažurirati na pretpostavljenu vrednost.

Ove opcije su navedene radi kompletnog opisa međurelacionih ograničenja u relacionom modelu podataka. Implementacija ovih ograničenja zavisi od konkretnog softvera za rukovanje bazom podataka.

3. Operativna komponenta

Kao što je opisano u odeljku 1.1 operativnu komponentu čine jezika [ML96]:

- **upitni jezik,**
- **jezik za ažuriranje podataka,**
- **jezik za definiciju podataka**.

U relacionom modelu podataka standardni jezik je SQL i on objedinjuje sve ove komponente. On poseduje visok nivo deklarativnosti što se ilustruje sledećim primerima.

Primer 2.8. Ako je potrebno prikazati definicije svih pojmova, koji su relevantni za nastavnu jedinicu *Električni kapacitet provodnika.Kondenzatori* iz prethodnog primera, onda bi realizacija upita imala sledeći oblik:

```
SELECT A.naziv, D.tekst
FROM Nastavne_jedinice A, Pojmovi_NasJed B, Dekompozicija C,
     Pojmovi D
WHERE A.sif_nasjed=B.sif_nasjed AND B.sif_pojma=C.sif_pojma1 AND
      C.sif_pojma2=D.sif_pojma AND
      A.naziv="Električni kapacitet provodnika.Kondenzatori"
```

Primer 2.9. Ako je potrebno prikazati ***sve pojmove*** koji učestvuju u dekompoziciji ***svih*** nastavnih jedinica realizacija ovog upita ima sledeći oblik:

```
SELECT A.sif_nasjed, A.naziv, D.tekst
FROM Nastavne_jedinice A, Pojmovi_NasJed B, Dekompozicija C,
     Pojmovi D
WHERE A.sif_nasjed=B.sif_nasjed AND B.sif_pojma=C.sif_pojma1 AND
      C.sif_pojma2=D.sif_pojma
GROUP BY A.sif_nasjed
```

Dekompozicija pojmova upotrebom objektno-orijentisanog modela podataka

Kao što je rečeno u odeljku 1.1.4., objektno - orijentisani model podataka nije jasno teorijski definisan i ne postoji standardni objektno - orijentisani model. U [ML96] definisani su koncepti ovog modela, po kome se vrši projektovanje baza podataka u nastavi za slučaj dekompozicije pojmova.

1. Strukturalna komponenta

- **objekat**

Entitete nastavnog gradiva (oblasti, jedinice, teme) moguće je predstaviti putem objekata, pri čemu je moguće definisati i njihovu strukturu i skup metoda (procedura) za svaki entitet.

Primer 2.10. Ako je u bazi podataka obrazovnog softvera potrebno predstaviti ***didaktičke celine*** u obliku teksta, slike i animacije sledeća linearna struktura podataka može predstavljati *strukturalni deo* odgovarajućeg objekta:

((IME,*Električni kapacitet provodnika*), (TEKST, *Električni kapacitet provodnika je...*), (SLIKA, *BMP fajl*), (ANIMACIJA, *AVI fajl*)),

dok skup

{*prikaži_podatke, dodaj_novu_jed, dekompozicija*}

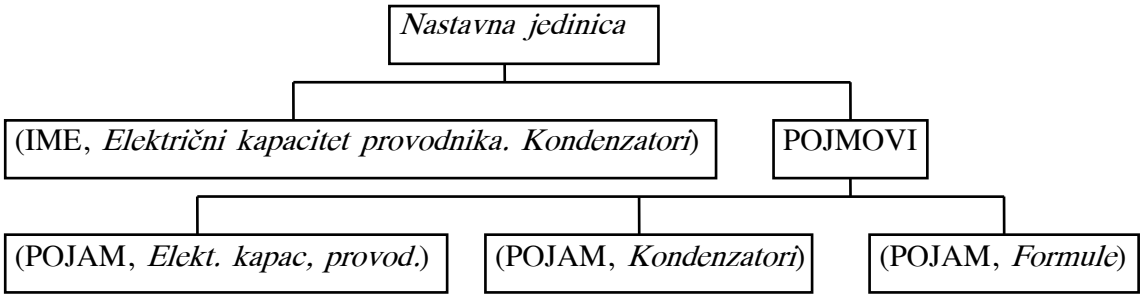
sadrži nazive *metoda* koje se mogu primeniti na datu strukturu podataka.

Međutim, objekat može osim jednostavne strukture imati i složeniju strukturu, na primer, ***strukturu tipa stabla***, putem koje se realizuje dekompozicija pojmova.

Primer 2.11. Definisan je objekat *Nastavna_jedinica* sa sledećom strukturom:

((IME, *Električni kapacitet provodnika.Kondenzatori*), (POJMOVI, {(POJAM, *Električni kapacitet provodnika*), (POJAM, *Kondenzatori*), (POJAM, *Formule*)}))

Ovaj složeni objekat može se predstaviti sledećom strukturom tipa stabla (slika 2.4.):



Slika 2.4.

- **klasa**

U objektno-orijentisanom modelu podataka elementi domena obeležja objekata mogu uzimati vrednosti iz skupa vrednosti nekog osnovnog tipa podataka (Chr, Num, Date), ali i iz skupa objekata složenije strukture (tip torke ili skupa). S obzirom da su i elementi osnovnih tipova podataka objekti, sa svojom strukturom i ponašanjem, zaključuje se da domen predstavlja klasu.

Primer 2.12. Neka su definisani sledeći tipovi objekata:

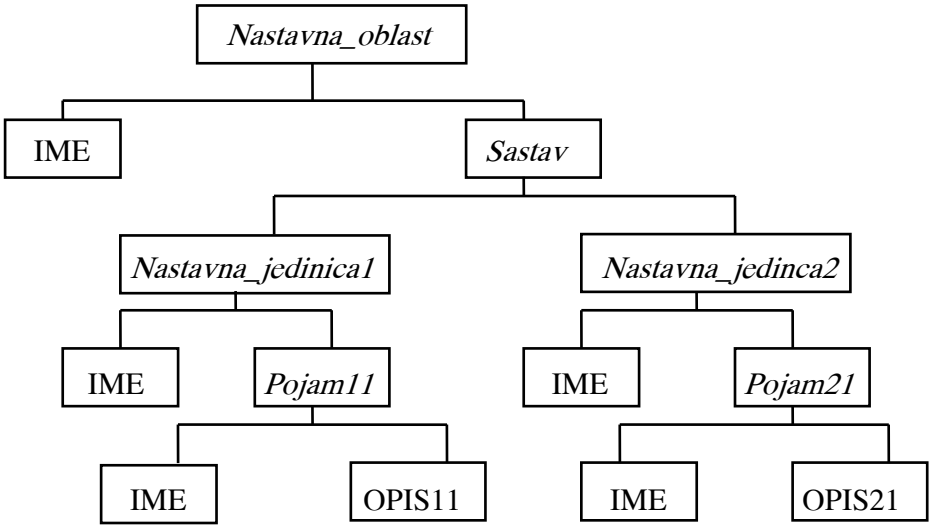
Pojam = ((IME, *Chr*), (OPIS, *Chr*))
Nastavna_jedinica = ((IME, *Chr*), (SASTAV, *Pojam*))

Polazeći od prethodne definicije pojma **klasa**, može se reći da u komponenti (SASTAV, *Pojam*) tipa objekta *Nastavna_jedinica* SASTAV je obeležje a *Pojam* je klasa. Vrednosti obeležja SASTAV su pojave klase *Pojam*.

Na sličan način je moguće definisati i tip objekta *Nastavna_oblast*:

Nastavna_oblast = ((IME, *Chr*), (SASTAV, *Nastavna_jedinica*))

Na slici 2.5. objekat *Nastavna_oblast* prikazan je kao struktura nad skupom objekata. Listovi stabla sadrže primitivne objekte, dok drugi čvorovi predstavljaju agregate (složene objekte), koji su na slici naznačeni samo svojim opštim imenom.



Slika 2.5.

U sistemima za rukovanje bazama podataka zasnovanim na relacionom modelu obezbeđeno je rukovanje ekstenzijom. To se postiže definisanjem *tipa* odnosno koncept šeme relacije omogućuje upravljanje njenom ekstenzijom (svim mogućim njenim pojavama). U objektno-orijentisanom modelu podataka ne postoji mogućnost upravljanja svim pojavama neke klase. To je moguće samo na osnovu identifikatora objekata. Identifikator objekata ne odgovora pojmu ključa u relacionom modelu podataka, nego je on pod kontrolom samog softvera za rukovanje bazama podataka.

- **nasleđivanje**

Koncept nasleđivanja objektno-orijentisanog modelu podataka je važan za projektovanje nastavne baze podataka, jer definiše odnos ***"je podvrsta"*** među klasama. Putem ovog odnosa se predstavlja hijerarhijska struktura nastavnog gradiva.

Primer 2.13. Ovaj koncept je objašnjen na primeru definisanja klase *NastavnaOblast* i njene potklase *NastavnaJedinica*. U ovom primeru korišćena je sintaksa objektno - orijentisanog jezika C++.

```
class NastavnaOblast{
        tip_podatka_koji_sadrži_ime_oblasti IME;
public:
        funkcija PrikaziIme();
        funkcija PrikaziStrukturu();
};
//Definicija funkcije
//...
funkcija NastavnaOblast::PrikaziIme() {
        Ispiši ime nastavne oblasti u rečenici "Nastavna oblast se zove ...".
}
//...
class NastavnaJedinica : public NastavnaOblast {

public:
        NastavnaJedinica();
        ~NastavnaJedinica();
        void PrikaziPojmove();
}
```

To znači da će objekti klase *NastavnaJedinica* automatski naslediti obeležje IME i sve metode: *PrikaziIme*, *PrikaziStrukturu*. Ako neki objekat klase *NastavnaJedinica* primi poruku *PrikaziIme*, on će odgovoriti na nju tako što će se pozajmiti odgovarajuća metoda od klase *NastavnaOblast*. Kao što se vidi, objektu klase *NastavnaJedinica* dodata je metoda *PrikaziPojmove*.

2. Integritetna komponenta

- **identitet objekta**

U objektno - orijentisanom modelu postoji nekoliko načina za realizaciju identiteta objekta [ML96]:

- virtuelne i fizičke adrese,
- korisnički nazivi i
- surogati.

Objekti su jedinstveno identifikovani putem vrednosti ***sistemski*** definisanih atributa - identifikatora objekata (OID) i sastoje se od identifikacionog broja baze podataka, identifikacionog broja klase i identifikacionog broja instance, kao što je to slučaj u [IYI96]. Moguće je da identifikator objekta ima logičku prezentaciju i da nema fizičku adresu u sekundarnoj memoriji [IYI96]. U objektno-orijentisanom softveru za rukovanje bazama podataka O2 identifikator ima i fizičku adresu.

U slučaju dekompozicije pojmova, ova karakteristika objektno-orijentisanog modela podataka znači da identifikaciju nastavnih tema, jedinica i pojmova realizuje sam softver za rukovanje bazama podataka putem vrednosti sistemski definisanih atributa i stoga vrednost identifikatora nije podložna promenama od strane korisnika (projektanta ili administratora baze podataka).

3. Operativna komponenta

Objektno-orijentisani model podataka, pored nedovoljno jasno definisanih koncepata nema ni jasno definisane jezike za manipulaciju podacima i za definisanje podataka, jer se još uvek posmatra ili kao nadogradnja nad objektno-orijentisane programske jezike ili nadogradnja nad relacione softvere za rukovanje bazama podataka.

Primer 2.14. U [IYI96] definisan je upitni jezik kao integracija programskog jezika opšte namene (C) i jezika baza podataka u objektno-orijentisanom kontekstu, što je detaljnije opisano u odeljku 1.1.4. Upit kojim se realizuje prikaz imena pojmova, koji su relevantni za nastavnu jedinicu *Električni kapacitet provodnika* u ovom softveru ima sledeći izgled:

```
NastavnaJedinica.PrikaziPojmove()
 where NastavnaJedinica.ime==“Električni kapacitet provodnika”
```

Na ovom primeru se vidi da objektni izraz u upitu može da sadrži i metode objekata.

2.3.2. Primena logičkog modela podataka za proveru učeničkih hipoteza u dijaloškim sistemima

Sistemi automatskog rezonovanja (SAR) nisu isprva bili razvijeni za potrebe obrazovanja, ali se mogu iskoristiti za unapređenje rada učenika sa računarom. Razvoj sistema automatskog dokazivanja teorema (ADT), PROLOG-jezika i ekspertnih sistema omogućio je primenu SAR i u obrazovanju.

Opšta procedura rezonovanja dopušta upravljanje radom računara na višem nivou. To znači da programer i projektant ne moraju da predvide sve moguće opcije rada računara.
Sa druge strane, univerzalnost rezolucijske procedure čini sistem nezavisnim od problemske oblasti.

U [Hot95a] navedeno je nekoliko mogućnosti za razvoj inteligentnih svojstava dijaloga učenika i računara uz primenu SAR. To su sledeće mogućnosti:

- dobijanje odgovora na učenikovo pitanje (SAR dedukuje odgovor na osnovu postojećih činjenica),
- provera učenikovih hipoteza,
- otkrivanje protivrečnosti u izjavama učenika.

U prvom slučaju, u bazi znanja nalaze se opšta i specifična znanja. Učenik može da unosi i nova relevantna znanja u vezi sa datom problemskom oblašću. Učenik zadaje pitanje na koje sistem odgovara dedukujući odgovor na osnovu znanja i činjenica, koje postoje u bazi znanja. Mogući odgovori sistema su da je odgovor pronađen, odgovor nije mogao biti pronađen ili sistem traži od učenika dodatne informacije.

Primer 2.15. Zadatak je formulisan u [Hot95a]. Neka se u bazi podataka nalaze podaci o rekama i pritokama za određeni region u obliku sledeće šeme relacije ULIVA({REKA1,REKA2},{REKA1,REKA2}) sa značenjem da se REKA1 uliva u REKU2.
Torke ove relacije predstavljane su u BASELOG-sistemu (opisanom u odeljku 4.) putem aksioma tipa BAKS:
BAKS[1]:='Morava,Dunav'
BAKS[2]:='Sava,Dunav'
Ako učenik postavi pitanje "Gde se uliva Timok?" odnosno:

~uliva(Timok,V1)&

a u bazi podataka nema takve informacije, sistem može da traži od učenika da kaže šta zna o Timoku. Učenik može da unese informaciju: Timok se uliva u istu reku kao i Morava. Ovo pravilo ima u BASELOG-sistemu formu sopstvene aksiome:

AKS[1]:='~uliva(Morava,Z1)uliva(Timok,Z1)&'

Sistem automatskog rezonovanja na osnovu podataka o Moravi ("Morava se uliva u Dunav") dedukuje odgovor da se Timok uliva u Dunav. Zadatak je realizovan u [Ber97] upotrebom opisnog jezika logičkog programiranja.

Ovaj primer ilustruje situaciju u kojoj se upotrebom sistema automatskog rezonovanja može dedukovati činjenica koja se ne nalazi u bazi podataka, odnosno, u odnosu na klasične baze podataka pristup upotrebom SAR znači mogućnost dobijanja odgovora na upite, čije se komponente ne nalaze u bazi podataka, već se odgovor dedukuje na osnovu činjenica i znanja, automatskim logičkim rasuđivanjem.

U drugom slučaju, rad sistema se zasniva na proveri hipoteze (tvrđenja) učenika, koju sistem tretira kao teoremu, koju treba da dokaže na osnovu aksioma u bazi znanja. Mogući odgovori sistema su: hipoteza je tačna, hipoteza nije tačna i sistem ne može da potvrdi niti da odbaci hipotezu (zbog poluodlučivosti ili zbog ograničenja prostorno - vremenskih resursa).

Primer 2.16. Zadatak je formulisan u [Hot95a]. U ovom slučaju se radi o gradivu istorije. Učenik postavlja hipotezu: "Boj na Čegru je bio posle boja na Marici."

U bazi podataka se podaci nalaze u relaciji BOJ({MESTO,GODINA},{MESTO}), sa sledećom pojavom:

mesto	godina
Marica	1371
Kosovo	1389

(pri čemu u pojavi nad relacijom BOJ ne postoji podatak o tome kad je bio boj na Čegru);

i postoji šema relacije POSLE({BOJ,BOJ}, {BOJ,BOJ}) sa sledećom pojavom:

boj	boj
Cegar	Kosovo

Definišu se i pravila:

- boj na **X1** je bio posle boja na **Z1** ako je boj na **X1** bio godine **U1**, a boj na **Z1** godine **U2**, pri čemu je **U1 > U2**.
- ako je **X1** posle **Y1** i **Y1** posle **Z1** onda je **X1** posle **Z1.**

Sistem automatskog rezonovanja će na osnovu ovih podataka u bazi i na osnovu datih pravila potvrditi učenikovu hipotezu.

U opštem slučaju, sistem može da potvrdi, ne potvrdi ili odbaci učenikovu hipotezu.

U trećem slučaju, otkrivanje protivrečnosti u izjavama učenika, potrebno je da postoji baza znanja u vezi sa određenom problemskom oblašću a sistem pokušava da izvede protivrečnost, odnosno da dokaže da negacija izjave logički sledi iz kreirane baze znanja. Mogući odgovori sistema su: izjava učenika nije tačna (protivreči bazi znanja), izjava učenika je tačna (negacija izjave protivreči bazi znanja) i izjava ne protivreči bazi znanja, ali to ne znači da je izjava tačna.

Primer 2.17. Zadatak je formulisan u [Hot95a] i realizovan je na opisnom jeziku logičkog programiranja u [Ber97]. Pokazaće se kako se realizacija ovog zadatka može ostvariti u BASELOG-sistemu.

Neka je definisana sledeća šema relacije:

DELO({IME,PREZIME,NAZIV}, {NAZIV})
sa značenjem da je autor sa datim *imenom* i *prezimenom* napisao delo sa datim *nazivom*.

Pojava nad ovom šemom relacije je sledeća relacija:

ime	prezime	naziv
Laza	Lazarevic	Vetar
Laza	Lazarevic	Verter
Laza	Lazarevic	Na bunaru
Laza	Lazarevic	Svabica
Ivo	Andric	Na Drini cuprija
Ivo	Andric	Travnicka hronika
Ivo	Andric	Gospodjica

U BASELOG-sistemu se putem aksioma tipa BAKS predstavljaju torke relacije DELO na sledeći način:

BAKS[1]:='delo(Laza,Lazarevic,Vetar)&'
BAKS[2]:='delo(Laza,Lazarevic,Verter)&'
.
.
BAKS[7]:='delo(Ivo,Andric,Gospodjica)&'

Ako učenik navede da je Laza Lazarević autor dela "Gospodjica" sistem će otkriti protivrečnost. U formi BASELOG-sistema polazi se od negacije tvrđenja, što znači da izjava učenika ima oblik:

delo(Laza,Lazarevic,Gospodjica)&

a sistem daje odgovor:
uspeh=1
dokaz je nađen

što znači da je dokazana negacija te izjave i da ova izjava učenika nije tačna.

Sprega SAR sa nastavnim bazama podataka doprinosi povećanju stepena slobode korisnika u dijalogu sa računarom i pospešuje učenikovo rezonovanje pri postavljanju zahteva.

Ovo se naročito odnosi na nastavno gradivo koje je faktografskog karaktera (istorija, geografija, književnost), gde je učenikovo rezonovanje potisnuto pa ga treba razvijati upotrebom ovakvih sistema.

Logički model podataka omogućava projektovanje upitno-dijaloških sistema putem svojih komponenti što je ilustrovano sledećim kratkim primerima.

1. Strukturalna komponenta

- **Termi -** promenljive i konstante.
  Konstante su vrednosti obeležja šeme relacije.
  Promenljive pružaju mogućnost davanja odgovora na ciljeve, koji nisu u potpunosti instancirani konstantama.
  Na primer. Upit oblika:
  delo(Ivo, Andric, X)
  omogućuje izdvajanje svih dela Ive Andrića čiji se nazivi nalaze memorisani u bazi podataka.

- **Predikatske formule** - u pitanju su zatvorene formule (sve promenljive koje se javljaju u formulama su univerzalno kvantifikovane)

- **Torke relacija u bazi podataka** - putem torki se mogu predstaviti činjenice u bazi znanja
  Na primer, činjenica da je Ivo Andrić autor dela "Gospođica" predstavljena je putem aksiome tipa BAKS BASELOG-sistema na sledeći način:

  BAKS[7]:='delo(Ivo,Andric,Gospodjica)&'

- **Klauzule (sastavci)** - oblik kojim se definišu pravila (uslovi)
  Na primer, znanje da je boj na **X1** je bio posle boja na **Z1** ako je boj na **X1** bio godine **U1**, a boj na **Z1** godine **U2**, pri čemu je **U1 > U2** izraženo je na sledeći način [Berković I., Deduktivne osnove za razvoj opisnih jezika logičkog programiranja]:

  ```
  posle(X1,Z1) :- boj(X1,U1),boj(Z1,U2), >(U1,U2).
  posle(X1,Z1) :- posle(X1,Y1),posle(Y1,Z1).
  ```

- **Mehanizam zaključivanja** - rezolucijsko pobijanje, koje poseduje teorijsku univerzalnost odnosno teorijsku snagu za izvođenje svakog zaključka kad god je on logička posledica datih pretpostavki

- **Strategije** - od mogućih strategija (u širinu, u dubinu, kombinovana) u [Ber94] pokazano je da je dubinska strategija efikasnija od širinske.

2. Integritetna komponenta

Integritetnom komponentom logičkog modela podataka se zadaju **pravila** u postupku projektovanja nastavne baze podataka i dijaloških sistema putem kojih se želi ostvariti sledeće:

- učenik treba da dobije pedagoški prihvatljiv odgovor na upit,
- odgovor će biti pronađen (sprečiti mogućnost generisanja beskonačnih grana u stablu pretrage).

Integritetnu komponentu logičkog modela podataka čine:

- **Princip otvorenog i zatvorenog sveta** - putem deklarisanja liste CWA-predikata moguće je u BASELOG-sistemu definisati za koje će predikate sistem raditi u režimu otvorenog, zatvorenog ili poluotvorenog sveta.

Primer 2.18. Definisana je baza podataka u kojoj se nalaze samo sledeći elementi:

TROUGAO(zbir_unutrasnjih_uglova, 180)
KVADRAT(zbir_unutrasnjih_uglova, 360)
PRAVOUGAONIK(povrsina, ab)
KRUG(obim, $2r\pi$)

Neka su CWA-predikati: TROUGAO i KRUG.

Ako učenik postavi pitanje:

?- krug(povrsina, $r^2\pi$)

sistem će dati odgovor učeniku "**nisu**", jer je za predikat KRUG BASELOG-sistem radio u režimu zatvorenog sveta. Ovaj odgovor se ne sme dopustiti, jer dezinformiše učenika.□

Uzimajući u obzir semantiku nastavne baze podataka, kao i moguće upite učenika, potrebno je definisati listu CWA-predikata na taj način da BASELOG-sistem daje tačne i pedagoški prihvatljive odgovore.

- **Sigurnost pravila i izvođenja zaključaka** - putem liste CWA-predikata moguće je definisati rad sistema u režimu zatvorenog sveta, odnosno moguć je odgovor koji se dobija kao i u radu sa klasičnim bazama podataka (podatak postoji u bazi ili ne). U nekim situacijama se bez definisanja CWA-predikata odgovor na upit (dokaz tvrđenja) i ne može dobiti. Definisanjem dužine rezolvente i maksimalne dubine pretrage, takođe je moguće uticati na ishod (rezultat) dokaza.

3. Operativna komponenta

- **Jezik predikatskog računa** - u BASELOG-sistemu postoje sledeći koncepti operativne komponente:

- **sastavci** - putem ovog koncepta definišu se pravila
  Na primer, pravilo da ako su prave **X** i **Y** paralelne i prave **Y** i **Z** paralelne, onda su paralelne prave **X** i **Z** u implikativnoj formi ima oblik:
  $$paralelno(X,Y) \wedge paralelno(Y,Z) \Rightarrow paralelno(X,Z)$$
  odnosno, u formi sastavaka:
  ~PARALELNO(X1,Y1)~PARALELNO(Y1,Z1)PARALELNO(X1,Z1)
- **CWA-predikati** - definisanje izbora režima rada sistema
- **bazne aksiome** - predstavljanje torki iz relacija baze podataka. Svaka torka dobija status aksiome.

2.3.3. Kreiranje (projektovanje) modela korisnika

U postupku projektovanja modela korisnika je izvršena kombinacija relacionog i logičkog modela podataka. Tip obrazovnog softvera, koji u sebi ima model korisnika treba da omogući izvođenje zaključaka u odnosu na korisnika:

- koje informacije treba dati korisniku u narednim fazama učenja, ili
- koje sledeće pitanje postaviti u narednom koraku testiranja (vežbanja).

Ove informacije su karakteristične za tip softvera: *inteligentni tutorski sistemi*. Model korisnika se u literaturi zove i *studentov model* [AT91], [JD96], [El84].

Studentov model reprezentuje studentovo tekuće razumevanje sadržaja [AT91]. Tradicionalni CBI (Computer Based Instruction) programi imaju jednostranu reprezentaciju studentovog znanja (razumevanja) koje je tipično u obliku odgovora na pitanja (tačnih ili netačnih). Studentov model bi trebalo da ima više od jednostavnog skupa studentovih odgovora (njegovog rada), on bi trebalo da *memoriše i koje aspekte* sadržaja student razume (definicije, koncepte, pravila, principe, veze) i po mogućstvu kako ih je razumeo. Model treba konstantno da se nadopunjuje na osnovu novih informacija, zbog toga što inicijalni zaključci mogu biti pogrešni, i zbog toga što se studentovo razumevanje (napredovanje) menja.

Studentov model se primarno sastoji iz odluka (odgovora) kreiranih putem pravila, koja su ugrađena u *pedagoškom modelu*. Pedagoški model je druga komponenta softvera za učenje uz pomoć kompjutera i na ovom mestu se navodi zbog potrebe potpunijeg opisa studentovog modela. Elementi studentovog modela, koji reprezentuju nekompletno ili nepravilno razumevanje, su u vezi sa pravilima za davanje pomoći učeniku, pojašnjavanje i ponavljanje. Elementi studentovog modela, koji reprezentuju kompletno ili korektno razumevanje, prouzrukuju aktiviranje pravila za povećanje kompleksnosti, prelaz na nove teme, završetak lekcije, itd.

Pedagoški model sadrži pravila za učenje u opštijoj formi nego tradicionalni programi za učenje uz pomoć kompjutera.

Primeri opštih upravljačkih pedagoških pravila:

Ako student pravi greške, proveriti potrebno znanje.
Ako student ne odgovora, pitati ga da li je zbunjen.
Ako student pravi mnogo grešaka i radi više od 30 minuta, predložiti prekid.
Ako student radi odlično, preći na teže gradivo.

Specifična pravila su:

Ako student ne odgovara tačno na pitanje, pitati ga da ponovi pitanje da se vidi da li ga je razumeo.
Ako student reši neki problem nekorektno, tražiti da se odredi broj negativnih i pozitivnih poena (ili broj bodova).

Model sadržaja je treća komponenta CBI programa i on je u kombinaciji sa studentovim modelom. To je optimalna reprezentacija sadržaja. To su obično činjenice u bazi podataka, veze, i organizacija nad njima (kao što je objašnjeno u **odeljku 2.2. *Projekcija mogućnosti rešenja u domenu baza na ciljeve i zahteve obrazovnog računarskog softvera***)

Struktura nastavnog sadržaja može biti:
- hijerarhijska
- mrežna.

Hijerarhijska struktura može biti prezentovana putem relacija, kao što je pokazano u odeljku 2.3.1. Mrežna struktura može biti prezentovana u obliku matrice, gde redovi i kolone prezentuju informacije a ćelije predstavljaju vezu između informacija.

Na osnovu datog predstavljanja znanja u bazi, studentov model sadrži sličnu formu ali različitog sadržaja.

Studentov model nisu samo informacije o studentu. On uključuje program koji operiše nad bazom podataka, modifikujući je na osnovu studentovog rada da bi reflektovao (putem zaključivanja) da li je student nešto razumeo ili nije.

Na osnovu svega rečenog, znači da je u oblikovanju studentovog modela poželjno iskoristiti logički model podataka (naravno u kombinaciji sa relacionim).

Putem relacionog modela moguće je predstaviti podatke o učeniku (studentu). To je moguće, takođe i upotrebom BASELOG-sistema, gde se putem aksioma tipa BAKS predstavljaju torke relacione baze podataka. U BASELOG-sistemu je moguće putem sopstvenih aksioma predstaviti *pedagoški model*, odnosno pravila zaključivanja (ako je korisnik dao tačan odgovor dati mu novo (teže) pitanje, ako nije - vratiti ga na staro ili mu dati help ili mu predložiti neki drugi oblik učenja).

Primer 2.19. Realizacija studentovog modela na opisnom jeziku logičkog programiranja.

Neka su u pedagoškom modelu definisana sledeća pravila:

1) *Ako student ne odgovora, pitati ga da li je zbunjen .*
2) *Ako student pravi mnogo grešaka i radi više od 30 minuta, predložiti prekid.*
3) *Ako student radi odlično, preći na teže gradivo.*

Neka je u bazi podataka definisana sledeća šema relacije:
UČENIK({id_br_uc, ime, prezime, id_br_pit, odgovor, vreme},{id_br_uc})

Neka su obeležjima *odgovor* i *vreme* pridruženi sledeći domeni:
dom(*odgovor*)={prazno, netacno, tacno}
dom(*vreme*)={0..45}

Jedna pojava nad ovom šemom relacije ima sledeći izgled:

id_br_uc	ime	prezime	id_br_pit	odgovor	vreme
001	Jovan	Simic	10	tacno	15
001	Jovan	Simic	11	netacno	20
002	Nikola	Peric	10	0	30
002	Nikola	Peric	13	tacno	25
003	Mira	Tot	10	tacno	45
...	...	...	...	...	

U opisnom jeziku logičkog progamiranja, opisanom u [Ber97], pedagoška pravila mogu biti formulisana na sledeći način.

- Sledeće pravilo definiše prikaz imena i prezimena učenika koji nije uopšte odgovorio na postavljeno pitanje. Argument sa vrednošću 0 predikata *ucenik* je vrednost obeležja *odgovor* istoimene relacije iz baze podataka.

```
postavi_pitanje_uceniku(X1,X2) :- ucenik(X1,X2,U1,0,Z1).        (1)
```

Rezultat ovog upita nad podacima u relaciji UČENIK je:

zamena: Nikola/X1;Peric/X2;10/U1;30/Z1

Izraz "zamena: Nikola/X1" znači da se vrši supstitucija promenljive X1 vrednošću Nikola.

Dobijeno je ime i prezime učenika koji nije odgovorio na pitanje, pa je potrebno preduzeti adekvatnu akciju.

- U sledećem pravilu je definisana selekcija imena i prezimena učenika, kojima je za odgovor na dato pitanje trebalo više od 30 minuta. Promenljiva V1 će biti instancirana konkretnom vrednošću obeležja *vreme* relacije *učenik* koja je u pravilu broj (2) predstavljena putem istoimenog predikata.

predlozi_prekid(X1,X2) :- ucenik(X1,X2,U1,Z1,V1), >(V1,30). (2)

Rezultat upita nad podacima u relaciji UČENIK je:

zamena: Mira/X1;Tot/X2,10/U1,netacno/Z1,45/V1

Dobijeno je ime i prezime učenika, kome treba predložiti prekid rada, jer je tom učeniku trebalo više od 30 minuta da odgovori na dato pitanje.

- Sledeće pravilo, takođe, definiše selekciju imena i prezimena učenika, ali onih koji su tačno odgovorili na dato pitanje za vreme manje od 30 minuta:

predji_na_teze(X1,X2) :- ucenik(X1,X2,U1,"tacno",Z1), <(Z1,30) (3)

Rezultat upita nad podacima u relaciji UČENIK je:

zamena: Jovan/X1;Simic/X2,10/U1,tacno/Z1,15/V1
zamena: Nikola/X1;Peric/X2,13/U1,tacno/Z1,25/V1

Međutim, moguće je implementirati studentov model i u nekom od softvera za rukovanje bazama podataka sa aktivnom komponentom. To su POSTGRES i NAIL!

Primer 2.20. Definisati pravilo putem koga će se učeniku zadati određeno sledeće pitanje u zavisnosti od vrednosti datog odgovora. Za realizaciju prethodno spomenutog pravila, pored relacije:

UCENIK({id_br_uc, ime, prezime, id_br_pit, odgovor, vreme}, {id_br_uc})

u bazi podataka potrebno je da se formira i relacija:

PITANJA({id_br, tekst, id_br_sled_pit}, {id_br})
pri čemu je semantika obeležja *id_br_sled_pit* identifikacioni broj sledećeg pitanja ako je učenik tačno odgovorio na dato pitanje.

U sintaksi NAIL! softvera za rukovanje bazama podataka [Ullm88] realizacija se sastoji iz dva pravila. Prvim pravilom se izdvajaju ime i prezime učenika i identifikacioni broj pitanja na kojeg je učenik tačno odgovorio. Drugim pravilom se definiše prikaz imena i prezimena učenika i identifikacioni broj sledećeg pitanja. Pravila imaju oblik:

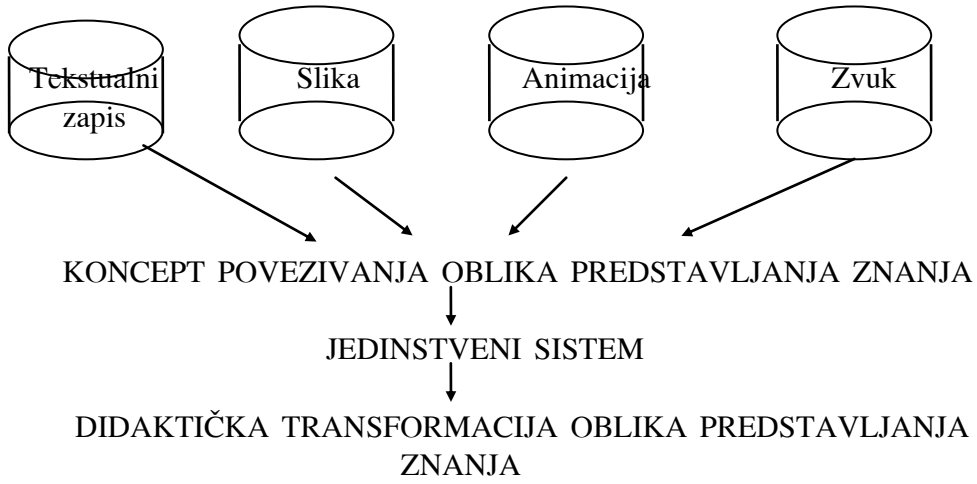
postavi_sled_pit(ime,prezime,id_br_pit) :- *ucenik*(ime,prezime,id_br_pit,"tacno",vreme) (1)

sled_pit(ime,prezime,id_br_sled_pit) :-
postavi_sled_pit(ime,prezime,id_br_pit),*pitanja*(id_br_pit,tekst,id_br_sled_pit) (2)

Putem pravila u aktivnim bazama podataka moguće je definisati ***prirodni spoj relacija***. Nakon primene pravila (1) dobiće se spisak imena učenika koji su tačno odgovorili na pitanje, čiji je identifikator *id_br_pit*. Primena pravila (2) se realizuje putem prirodnog spoja relacije *pitanja* i relacije, koja je rezultat primene prvog pravila *postavi_sled_pit*, a prirodni spoj je realizovan na osnovu vrednosti zajedničkog obeležja *id_br_pit*. Na taj način se za učenike, koji su tačno odgovorili na dato pitanje, dobija identifikacioni broj pitanja, koje će im se postaviti u narednom koraku učenja.

2.4. Integracija različitih vidova predstavljanja znanja u bazama podataka

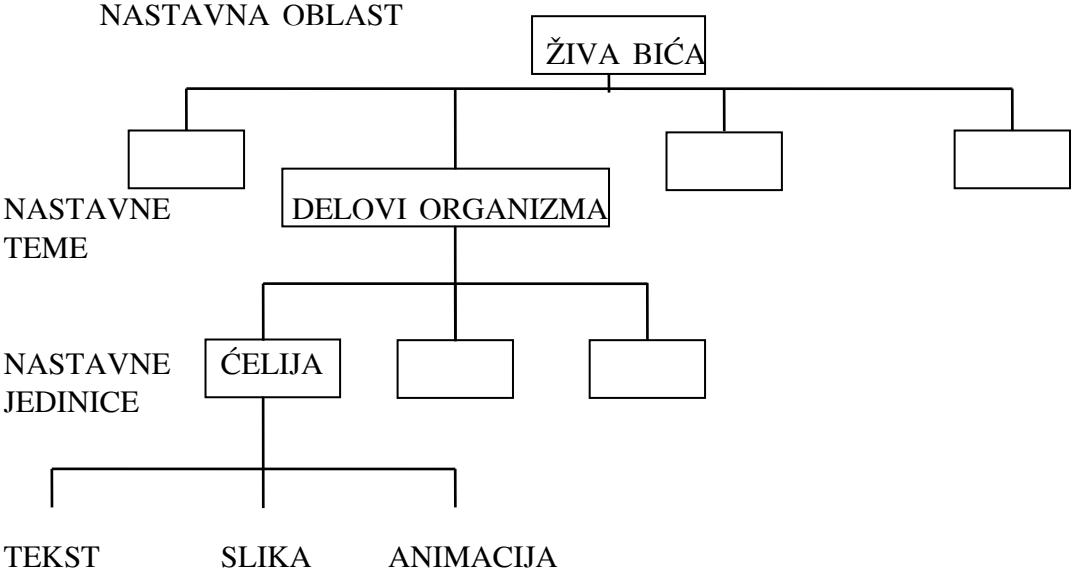
U ovom poglavlju definisan je koncept povezivanja različitih oblika predstavljanja znanja u cilju realizacije didaktičke transformacije znanja. Pojam didaktičke transformacije znanja objašnjen je u odeljku 2.1. Pretpostavka od koje se ovde polazi, je da su oblici predstavljanja znanja za date nastavne pojmove memorisani u vidu nekog od odabranog koncepta modela podataka (relacije, objekti) u različitim bazama podataka što je ilustrovano na slici 2.6.



Slika 2.6.

Cilj ovakvog projektovanja baza podataka u nastavi i učenju je omogućiti selekciju oblika predstavljanja pojmova od strane učenika odnosno korisnika ovakve baze. Učenik treba da dobije tačnu informaciju o nekom pojmu, ali u takvom obliku koji je za njega prihvatljiv (odgovarajući). U [AT91] je istaknuto da je prezentacija sadržaja u obrazovnim softverima moguća u obliku hijerarhijske ili mrežne strukture. U odeljku 2.3.1. dat je primer hijerarhijske strukture pojmova koja je realizovana putem relacionog modela podataka. U ovom odeljku će se uzeti predstavljanje nastavnog sadržaja putem hijerarhijske strukture.

Primer 2.21. Za nastavnu oblast ***živa bića*** predstavljena je struktura gradiva na sledeći način (slika 2.7):



Slika 2.7.

Predstavljanje nastavnog gradiva u obliku hijerarhijske strukture pruža mogućnost za povezivanje različitih oblika predstavljanja istog pojma, jer je putem ove strukture moguće precizno definisati pripadnost svakog didaktičkog koncepta (pojam, tema, jedinica) datom nadređenom konceptu u hijerarhiji koncepata. Ovu hijerarhijsku strukturu moguće je predstaviti putem dijagrama ER modela podataka što ilustruje slika 2.8.

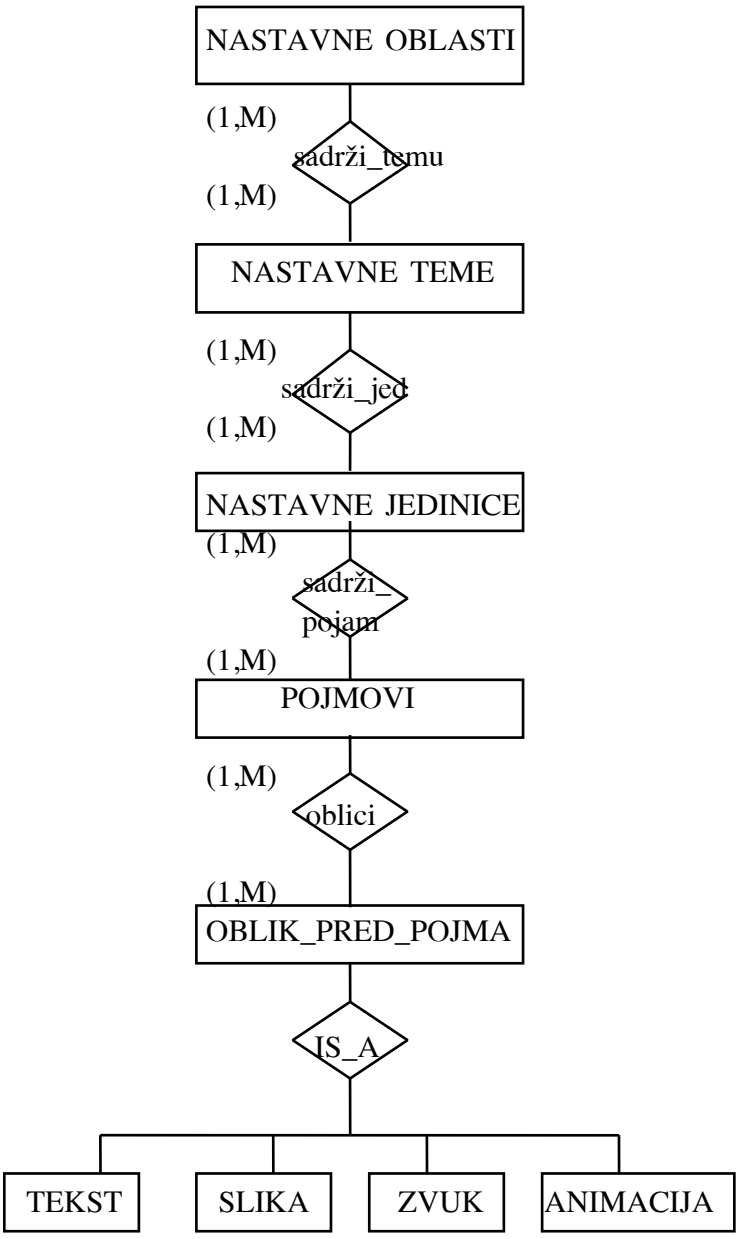
Putem IS_A hijerarhije definisan je odnos "*je podvrsta*" između oblika predstavljanja pojmova.
Saglasno pravilima za prevođenje ER modela u relacioni, ovaj ER dijagram se prevodi u sledeće šeme relacija:
nastavne_oblasti({ID_NO, *ostala_obeležja*}, {ID_NO})
nastavne_teme({ID_NT, *ostala_obeležja*}, {ID_NT})
nastavne_jedinice({ID_NasJed, *ostala_obeležja*}, {ID_NasJed})
pojam({ID_Pojma, *ostala_obeležja*}, {ID_Pojma})
oblik_pred_pojma({ID_Oblik_predstav})
tekst({ID_Oblik_predstav, tekst},{ID_Oblik_predstav})
slika({ID_Oblik_predstav, slika},{ID_Oblik_predstav})
animacija({ID_Oblik_predstav,animacija},{ID_Oblik_predstav})
zvuk({ID_Oblik_predstav,zvuk},{ID_Oblik_predstav})
sadrži_temu({ID_NO,ID_NT}, {ID_NO,ID_NT})
sadrži_jed({ID_NT, ID_NasJed}, {ID_NT, ID_NasJed})
sadrži_pojam({ID_NasJed, ID_Pojma}, {ID_NasJed, ID_Pojma})
oblik({ID_Pojma, ID_Oblik_predstav}, {ID_Pojma, ID_Oblik_predstav})

Primer 2.22. U bazi podataka nalaze se različiti oblici predstavljanja znanja memorisani u vidu ***n***-torki relacije (redova tabele). Relacije ***tekst, slika, animacija, zvuk*** sadrže konkretne oblike predstavljanja pojma. Svaka ***n***-torka ovih relacija ima identifikacioni broj kao primarni ključ a drugo obeležje predstavlja konkretni zapis (tekst, slika, zvuk), čiji format zavisi od samog softvera za rukovanje bazama podataka. [Nad97a]

Kada je potrebno objediniti sve oblike predstavljanja ***jednog*** pojma (na primer, ***ćelija***) prvo je potrebno odrediti vrednost njegovog primarnog ključa - ID_pojma. Ako je u relaciji ***oblik***, koja predstavlja pojavu tipa poveznika ***oblici*** ER dijagrama sa slike 2.8., pronađena vrednost datog ID_pojma, onda se za svaku tu vrednost određuje i vrednost primarnog ključa oblika predstavljanja pojma - ID_oblik_predstav. Nakon što su te vrednosti određene, sam oblik predstavljanja znanja datog pojma se pronalazi u odgovarajućim relacijama (***tekst, slika, animacija, zvuk***).
Realizacija opisanog konceptualnog rešenja povezivanja različitih oblika predstavljanja znanja moguća je upotrebom relacionih softvera za rukovanje bazama podataka. Mehanizmi putem kojih se to ostvaruje su:

- definisanje referencijalnih integriteta i
- definisanje veza među relacijama (tabelama).

Na taj način se veliki deo semantike iz procedura prenosi u mehanizme samih softvera za rukovanje bazama podataka.



Slika 2.8.

Pored ovih karakteristika, važno je napomenuti da softveri za rukovanje bazama podataka moraju imati mogućnost predstavljanja nestandardnih tipova podataka. Tehnike putem kojih se povezuju različiti oblici predstavljanja znanja opisane su u [Nad97a]. Opisani su softverski i hardverski zahtevi i specifičnosti alata za projektovanje multimedijalnih aplikacija. Navedene su karakteristike objektno-orijentisanog programiranja, koje se odnose na povezivanje slike, teksta, zvuka i video-animacija.

3. TEORIJSKE OSNOVE ZA REALIZACIJU DEDUKTIVNIH MODELA BAZA PODATAKA

3.1. Logički potpuni rezolucijski sistemi rezonovanja

U oblasti automatskog rezonovanja centralno mesto pripada deduktivnim sistemima. Deduktivni sistemi omogućuju potpuno automatsko dokazivanje teorema, kao i interaktivno dokazivanje teorema. Razlikuju se rezolucijski i nerezolucijski sistemi. Rezolucijski sistemi su zasnovani na pravilu rezolucije, dok nerezolucijski sistemi koriste razna pravila na bazi ideje prirodnog izvođenja. Privlačnost rezolucijskih sistema se ogleda u postojanju samo jednog pravila izvođenja čiju je logičku korektnost i potpunost dokazao J. Robinson 1965. godine [Ro65]. Tako je razvijena metoda rezolucije koja se koristi u ovom radu.

3.1.1. Metoda rezolucije i teorema o rezoluciji

Formalne teorije su okvir logičkih sistema u kojima se ostvaruje formalizacija dokazivanja teorema. Najpoznatije formalne teorije su: iskazni računi, kvantifikatorski računi, računi viših redova, modalne logike itd. Predikatski račun prvog reda je univerzalni jezik za prezentaciju znanja i rešavanje problema iz raznih domena.

Formalizacija zadatka na predikatskom računu prvog reda je potrebna da bi se realizovalo dokazivanje teoreme metodom rezolucije. Kada je reč o dokazivanju teorema, sve formule moraju biti zatvorene (sve promenljive koje se pojavljuju u formulama su univerzalno kvantifikovane). U tom slučaju važi sledeća veza formalnog izvođenja sa semantikom:

$$A1, A2,...,Ak \vdash A \text{ akko } \models (A1 \wedge A2 \wedge ... \wedge Ak) \Rightarrow A,$$

što ima sledeće značenje:

Zatvorena formula A je izvodiva iz konačnog skupa zatvorenih formula $F=\{A1,A2,...,Ak\}$ akko je formula $(A1 \wedge A2 \wedge ... \wedge Ak) \Rightarrow A$ valjana. Formula je valjana ako je tačna u svakoj interpretaciji. Ako postoji formalno izvođenje formule A iz skupa premisa F, kaže se da je A logička posledica skupa F. Sa druge strane, formula A je semantička posledica skupa F akko je formula A tačna u svakoj interpretaciji u kojoj je tačna svaka od formula skupa F. Ekvivalentnost sintaksne i semantičke koncepcije izražava stav potpunosti predikatskog računa [HK92]:

$$\vdash A \text{ akko } \models A$$

Formula A je teorema (formalno izvodiva iz aksioma predikatskog računa) akko je A valjana odnosno tačna u svakoj interpretaciji.

Ovaj stav omogućuje da se semantika izvođenja zaključka A iz premisa F zameni formalnim izvođenjem (dokazom) za A1,...,Ak $\vdash$ A.

Za valjane formule postoji mehanička procedura koja u konačnom broju koraka utvrđuje da je formula valjana, što daje osnovu da se predikatski račun smatra poluodlučivim.

Na osnovu stava potpunosti to znači da postoji procedura koja u konačnom broju koraka za teoremu A utvrđuje da je A teorema (logička posledica skupa formula F). Ako formula A nije teorema, takva procedura, u opštem slučaju, ne završava rad i ne daje odgovor. Upravo u tome se ogleda poluodlučivost predikatskog računa.

Rezolucijska metoda se zasniva na metodi pobijanja: izvođenje $\mathbf{F} \vdash \mathbf{A}$ može se zameniti pobijanjem $\mathbf{F} \cup \{\neg \mathbf{A}\}$, odnosno važi **stav** ([Hot95], str. 59):

A1, A2,...,Ak $\vdash$ A ako i samo ako A1,A2,...,Ak,$\neg$A $\vdash$ *protivrečnost.*

gde su A1, A2,..., Ak, A zatvorene formule predikatskog računa prvog reda.
Kaže se da je skup {A1,A2,...,Ak, $\neg$A}, iz koga se izvodi protivrečnost nezadovoljiv.

Znači, potrebno je negirati formulu koja se dokazuje i izvršiti odgovarajuću pripremu predikatskih formula (skolemizaciju i transformaciju u skup sastavaka [Hot95], [HP88]). Dobija se konačan skup sastavaka od kojih je svaki disjunkcija konačnog broja literala.

Pravilo rezolucije ima osobinu ***kompletnosti*** a to znači da se uz teoretsku pretpostavku neograničenih prostorno-vremenskih resursa može izvesti dokaz svake formule koja jeste teorema.

Za razumevanje metode rezolucije neophodni su pojmovi koji se navode prema [Hot95], [HP88]. Definicije pojmova: sastavak, literal, komplementaran par, zamenska komponenta, zamena, unifikator, najopštiji unifikator prethode formulisanju pravila rezolucije i teoreme o rezoluciji.

Elementarni pojmovi kao što su: pojam konstante, promenljive, operacije, funkcije, terma, formule i atomične formule, smatraju se poznatim i ovde se neće posebno definisati.

Definicija 3.1. Literal je atomična formula ili njena negacija.

Definicija 3.2. Sastavak je konačan skup literala. Prazan skup literala je ***prazan sastavak***.

Neprazan sastavak se može prikazati formulom oblika:

L1 $\vee$ L2 $\vee$ . . . $\vee$ Lm,

pri čemu je: $m \geq 1$, Li su literali a $\vee$ oznaka za disjunkciju.
U tom smislu, za $m \geq 2$ sastavak je disjunkcija literala.

Literali Li, za i=1,2,...,m imaju oblik:

$$P(t_1,t_2,...,t_n) \text{ ili } \neg P(t_1,t_2,...,t_n)$$

pri čemu je: P - predikatski simbol, t_i su termi (za i=1,2,...,n), a $\neg$ je simbol negacije.

Definicija 3.3. Ako je **A** ***atomična formula***, onda su literali **A** i **¬A** komplementarni i obrazuju ***komplementaran par***.

U procesu dokazivanja teorema veoma važnu ulogu ima ***unifikacija***. Njen zadatak je da ujednači (unificira) literale, sastavke, formule ili njihove skupove, pri čemu se zadržava što veći stepen slobode (maksimalno se čuvaju promenljive).

Definicija 3.4. Zamenska komponenta je zapis oblika: **t/x**, gde je **x** promenljiva a **t** term različit od **x.**

Zamena je konačan skup zamenskih komponenti:
$\theta = \{ t_1/x_1,...,t_n/x_n \}$, gde je $x_i \neq x_j$ za $i \neq j$.
Prazna zamena je prazan skup zamenskih komponenti.

Primena zamene θ na skup C, gde je C skup terma ili skup literala je skup Cθ određen jednovremenim zamenjivanjem promenljivih $\mathbf{x_i}$ termima $\mathbf{t_i}$, pri čemu je $\mathbf{t_i/x_i}$ zamenska komponenta zamene θ. Skup Cθ zove se θ-primer skupa C.

Definicija 3.5. Zamena θ je ***unifikator*** skupa C, gde je C skup terma ili skup literala, ako je Cθ jednočlan skup. Ako za skup C postoji unifikator θ, kaže se da θ unificira skup C, odnosno da je C unifikativan skup.

Definicija 3.6. Neka su θ i σ unifikatori skupa A. Unifikator σ je ***najopštiji unifikator*** (NOU) skupa A, ako za ma koji unifikator θ skupa A postoji zamena λ takva da je $\theta = \sigma\lambda$, gde je $\sigma\lambda$ kompozicija redom zamena σ i λ.

Pojam kompozicije zamena može se naći u [HP88] i [Hot95]. Manje formalno rečeno, najopštiji unifikator se može shvatiti kao unifikator koji "maksimalno čuva promenljive".

Postoji **algoritam unifikacije** koji određuje najopštiji unifikator (NOU) za unifikativan skup literala i daje odgovor o nepostojanju NOU kada skup literala nije unifikativan. [Hot95]

Definicija 3.7. Neka za sastavke D1 i D2, koji ne sadrže zajedničke promenljive (što se uvek može postići preoznačavanjem) i za L1 $\subseteq$ D1, L2 $\subseteq$ D2 važe uslovi:

(1) Skupovi L1 i L2 nisu prazni,
(2) Za skup A svih atomičnih formula sadržanih u L1 i L2 postoji NOU σ_A,
(3) Elementi jednočlanih skupova L1σ_A i L2σ_A obrazuju komplementaran par.

Rezolventa sastavaka D1 i D2 je sastavak: (D1\L1) $\sigma_A \cup$ (D2\L2) σ_A.

Posebno, kad su L1 i L2 jednočlani skupovi, odnosno, literali u sastavcima D1 i D2 oblika L1∨C1, odnosno L2∨C2, pri čemu je L2=¬L1, rezolventa je oblika C1∨C2.
Na osnovu ovog posebnog slučaja sagledava se karakter pravila rezolucije, koja je u opštem slučaju dokazana u [Ro65].

Definicija 3.8. (***Pravilo rezolucije***) Neka su D1 i D2 sastavci za koje su ispunjeni uslovi (1)-(3) definicije 3.7. i R njihova rezolventa. Pravilo izvođenja $\frac{D1,D2}{R}$ zove se ***rezolucija***.

Metoda rezolucije sastoji se u generisanju rezolventi polazeći od početnog skupa sastavaka S.

Nivoi generisanih skupova primenom pravila rezolucije na elemente skupa S određuju se na sledeći način:

1. $R_0(S) = S$;
2. $R_{n+1}(S) = R(R_n(S))$, $n \geq 0$,

gde je $R(R_n(S)) = R_n(S) \cup r(R_n(S))$ i $r(R_n(S))$ je skup svih rezolventi skupa $R_n(S)$.

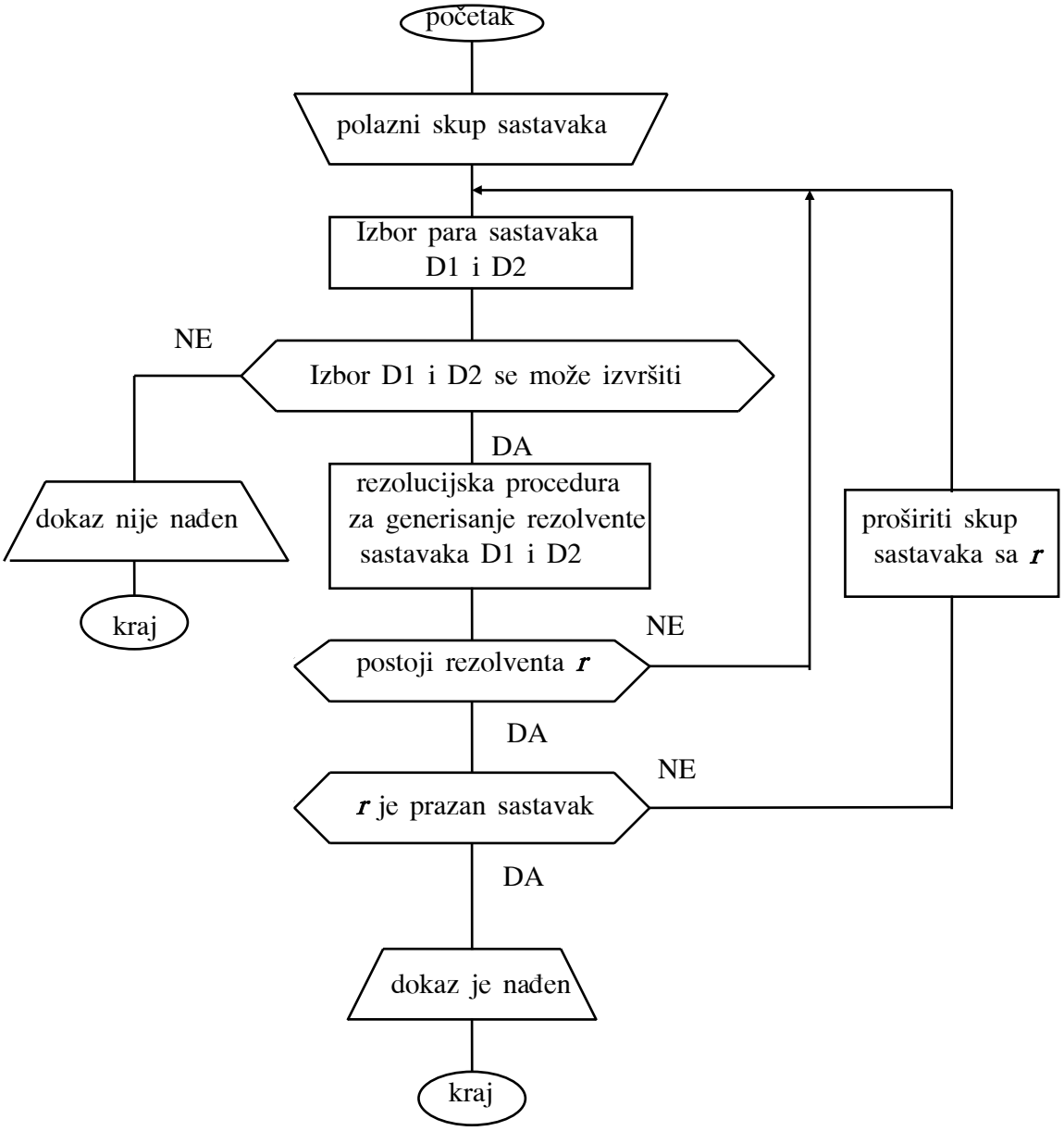
Teorema o rezoluciji [Ro65]:

Konačan skup sastavaka S je nezadovoljiv ako i samo ako skup $R_n(S)$ za neko konačno $n \geq 0$ sadrži prazan sastavak.

Procedura pobijanja koja se gradi neposredno na osnovu teoreme o rezoluciji ima više teorijski nego praktičan značaj, jer brojnost skupova $R_i(S)$ sa povećanjem i naglo raste.

Opšta shema takvih algoritama prikazana je na slici 3.1.

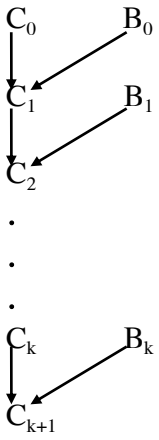
U praksi se primenjuju specifične forme rezolucije, koje poboljšavaju efikasnost procedure izvođenja. U [HP88] su opisane specifične forme rezolucije. Za potrebe ovog rada na osnovu [Ber97] i [Hot95] opisaće se uređena linearna rezolucija sa markiranim literalima.



Slika 3.1.

3.1.2. OL-rezolucija sa markiranim literalima

Uređena linearna rezolucija (OL) sa markiranim literalima kombinuje svojstva uređene i linearne rezolucije sa čuvanjem informacije o rezolviranim literalima. Struktura linearnog izvođenja se može prikazati sledećim usmerenim grafom (drvetom) na slici 3.2.:



Slika 3.2.

gde čvorovi C_i predstavljaju tzv. centralne sastavke, a čvorovi B_i predstavljaju tzv. bočne sastavke. C_o je početni centralni sastavak, koji pripada potpornom skupu sastavaka dobijenih iz negacije tvrđenja koje se dokazuje.

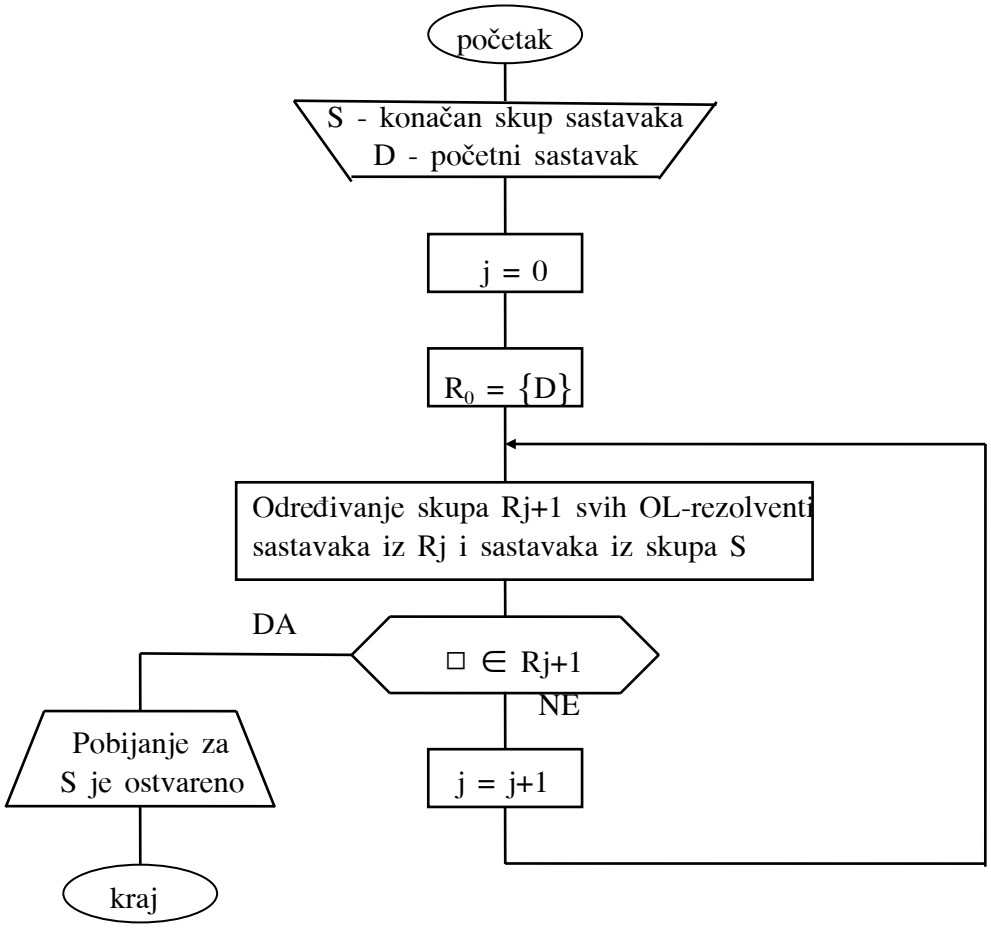
Centralni sastavak C_{i+1}, za $i \geq 0$, je OL-rezolventa centralnog sastavka C_i i bočnog sastavka B_i. Za bočne sastavke B_i uzimaju se sastavci iz početnog skupa sastavaka S.

Pored uvođenja uređenih sastavaka, koji se za razliku od sastavaka kao skupova literala, tretiraju kao uređene m-torke literala određene poretkom literala u sastavku s leva na desno, ova rezolucija čuva i informacije o rezolviranim literalima tako što ih na neki način markira. Njene prednosti, u odnosu na opštu rezoluciju su:

- dovoljno je memorisati samo rezolvente na uzastopnim nivoima, a ne sve rezolvente na svim nivoima pretrage,

- OL-rezolucija se vrši po poslednjem literalu centralnog sastavka, koji se markira i zadržava u OL-rezolventi i po k-tom (k=1,2,...) literalu bočnog sastavka. Na tako dobijenu rezolventu se primenjuju, ako je to moguće, operacije sažimanja i skraćivanja [Hot95].

Na slici 3.3. prikazana je shema algoritma za traženje pobijanja OL-rezolucijom. Oznake na šemi su: j - brojač niova, $\mathbf{R_{j+1}}$ je skup OL-rezolventi koje se obrazuju od sastavaka iz $\mathbf{R_j}$ (centralni) i sastavaka iz polaznog skupa S (bočni), • je oznaka za prazan sastavak ([Hot95], str. 81).



Slika 3.3.

Proces generisanja OL-rezolvente datog centralnog sastavka (d1) i datog bočnog sastavka (d2) može se opisati na sledeći način:

1. Preoznačavanje promenljivih (tako da u sastavcima ne bude zajedničkih promenljivih).

2. Određivanje najopštijeg unifikatora (NOU) θ za poslednji literal u sastavku d1 i k-ti literal (k=1,2,...) iz sastavka d2 (ako postoji za neko k, ako ne postoji onda se OL-rezolventa ne može odrediti).

3. Formiranje rezolvente markiranjem poslednjeg literala u d1θ i dopisivanjem ostatka sastavka d2θ bez k-tog literala (d1θ i d2θ su sastavci dobijeni primenom NOU θ na d1 i d2, respektivno).

4. Isključivanje identičnih nemarkiranih literala i ispitivanje tautologičnosti (tautologije se ne memorišu).

5. Operacija skraćivanja (brisanje svih markiranih literala iza kojih nema nemarkiranih).

6. Operacija sažimanja (brisanje poslednjeg nemarkiranog literala ako je komplementaran, u odosu na negaciju nekom markiranom literalu za unifikator λ).

7. Ponovna primena koraka: 5 i 6 sve dok je moguće (dok se na poslednji nemarkirani literal više ne bude mogla primeniti operacija sažimanja, ili dok se ne dobije prazan sastavak).

Ovim postupkom se formira jedna OL-rezolventa datog centralnog sastavka (d1) i datog bočnog sastavka (d2). Međutim, u slučaju da se rezolventa, koja je određena nakon koraka 5 ne memoriše, nego se memoriše samo finalna rezolventa nakon koraka 7, može doći do gubitka potpunosti metode. Koraci 1-5 predstavljaju ***ulaznu rezoluciju***, a koraci 6-7 ***rezoluciju prethodnika***.

U [HP88] na strani 57, a prema [Po76] važi ***potpunost*** OL-rezolucije u sledećem smislu:

Ako je sastavak D uređen sastavak u nezadovoljivom skupu S uređenih sastavaka i ako je S \ {D} zadovoljiv skup, onda postoji OL-izvođenje praznog sastavka iz S sa početnim sastavkom D.

3.1.3. Osnovne karakteristike ADT sistema

Sistem za automatsko dokazivanje teorema ADT sa varijabilnim strategijama omogućava variranje strategije pretraživanja uz korišćenje samo jedne vrste rezolucije: linearne uređene OL-rezolucije sa markiranim literalima. Ovaj sistem je razvila I. Berković [Ber94]. Sistem je razvijen na personalnom računaru i radi pod MS-DOS operativnim sistemom. Realizacija je izvršena na programskom jeziku Pascal (Turbo Pascal ver. 6.0 kompanije Borland International, Inc.). Sistem ADT poslužio je kao osnova za razvoj opisnih jezika logičkog programiranja [Ber97], inkorporiran je u sistem DEDUC za kreiranje kombinatornih rasporeda [Hot92], [Pro94]; koristi se u naučno-istraživačkom radu [LHRB95] i [LHRB97]; koristi se u nastavi.

Kao što je rečeno, sistem ADT omogućava variranje strategije pretraživanja. On raspolaže sa tri sintaksne strategije pretraživanja:

- strategija pregleda u širinu,
- strategija pregleda u dubinu i
- kombinovana strategija.

Širinskom strategijom pretraživanja generišu se sve moguće rezolvente ulaznom rezolucijom i rezolucijom prethodnika datog centralnog i svakog bočnog sastavka. Te rezolvente predstavljaju kandidate za centralne sastavke na sledećem nivou, itd.

Dubinskom strategijom se generiše samo jedna od tih rezolventi (prva rezolventa koja se može generisati) i ona postaje kandidat za centralni sastavak na sledećem nivou, itd.

Kombinovanom strategijom pretraživanja generišu se rezolvente širinskom strategijom dok ne dođe do popune nivoa. Zatim se, da ne bi došlo do gubitka rezolventi, vraća na prethodni nivo i pretraga nastavlja dubinskom strategijom.

U toku procesa dokazivanja nekom od raspoloživih strategija mogući su ishodi [Ber97]:

- za dati skup sastavaka nije moguće ostvariti pobijanje (to znači da je sistem-dokazivač završio rad i nije uspeo da generiše prazan sastavak: uspeh = 0, dokaz nije nađen),

- za dati skup sastavaka ostvareno je pobijanje (to znači da je dokazivač uspeo da generiše prazan sastavak: uspeh = 1, dokaz nađen),

- neizvestan ishod; u toku dokaza došlo je do prekoračenja zadatih parametara, što znači da se za dati skup sastavaka ne može zaključiti da li protivureči ili ne (to znači da su iscrpljeni svi raspoloživi resursi i rad nije okončan ni generisanjem praznog sastavka ni generisanjem svih mogućih rezolventi: uspeh = 2, neizvesno).

Centralni i bočni sastavci se u ADT sistemu čuvaju u nizovima KN[i], gde i=1,2,...,M i AKS[j], gde j=1,2,...,N, pri čemu M označava broj centralnih sastavaka, a N broj bočnih sastavaka. Centralni i bočni sastavci se unose u obliku stringova.

Sledi još nekoliko karakterističnih detalja rada ADT sistema:

- Sastavci se obavezno završavaju simbolom &.
- Oznaka za prazan sastavak je &.
- Disjunkcija između literala se ne piše.
- Oznaka za negaciju je ~.
- Oznaka za markirani literal je /.

Primer 3.1. Sastavak oblika

$$Q(A,X1) \vee \neg S(X1,Y1) \vee P(Y1,Z1)$$

gde su: Q - predikatski simbol, A - konstanta, X1,Y1,Z1 - promenljive,
u sistemu ADT se predstavlja kao:

Q(A,X1)~S(X1,Y1)P(Y1,Z1)&

U sistemu ADT se inicijalno zadaju sledeći parametri:
- maksimalna dužina sastavka,
- maksimalna dužina literala,
- maksimalna dužina rezolvente,
- maksimalna širina dokaza,
- maksimalna dubina dokaza,
- maksimalan broj centralnih sastavaka,
- maksimalan broj bočnih sastavaka.

3.2. PROLOG i konačan neuspeh

3.2.1. Osnovne karakteristike PROLOG-a

PROLOG je neproceduralni opisni jezik logičkog programiranja. Karakteristika neproceduralnih programskih jezika je da programer saopštava računaru **ŠTA** program treba da uradi a ne **KAKO** da reši problem. Modifikacijom sistema za rezolucijsko dokazivanje teorema, opisano u [Hot95], omogućen je razvoj deduktivnog sistema, koji predstavlja *proceduralnu osnovu* za razvoj PROLOG-a kao *deklarativnog jezika logičkog programiranja*.

Potrebno je uočiti razliku između *PROLOG-jezika* i *PROLOG-deduktivnog sistema*. PROLOG-sistem ima *proceduralni karakter* odnosno sastoji se od procedura koje određuju kako se realizuju izvođenja. PROLOG-jezik ima *deklarativni-karakter*, a to znači da se sistemu saopštava šta treba da radi, a ne kako. Sistemu se saopštava samo *opis* ciljeva i uslova, ali ne i način zadovoljenja ciljeva. Zbog toga je PROLOG-jezik ***opisni jezik***.

PROLOG-jezik je zasnovan na predikatskom računu prvog reda i pretpostavlja da su sve promenljive univerzalno kvantifikovane. Za opis problema koriste se Hornovi sastavci (klauzule). PROLOG-sistem koristi pravilo izvođenja - selektivnu linearnu rezoluciju definitnih klauzula sa negacijom shvaćenom kao konačni neuspeh **SLDNF** (Linear Resolution with Selection function for Definite clauses and Negation as Failure). [Bra86]

Karakteristične osobine PROLOG-a su (prema [Hot95]):
1. PROLOG-sistem je zasnovan na rezolucijskoj proceduri pobijanja ograničenoj na Hornove sastavke.
2. PROLOG-jezik sadrži tri vrste rečenica: ciljeve, pravila i činjenice.
3. Rezolucija se svodi na unifikaciju elemenata cilja sa nekom klauzulom (glavom te klauzule), pri čemu se generiše novi podcilj.
4. Polazni cilj će biti zadovoljen ako je PROLOG-sistem generisao prazan cilj (prazan sastavak).
5. Omogućeno je pronalaženje više različitih načina zadovoljenja cilja.
6. Podržana je primena rekurzije.

7. Procedura pretraživanja podržava vraćanje i rez.
8. PROLOG-programe je moguće tumačiti na dva načina: deklarativno i procedurno.
9. Jezik sadrži i neke karakteristike procedurnih jezika: rad sa listama i drugim strukturama, i proširen je aritmetičko-logičkim operacijama, ulazno-izlaznim naredbama i dr.
10. Negacija se u većini vrsta PROLOG-a tretira kao konačan neuspeh.

PROLOG se koristi u sledećim oblastima: ekspertni sistemi, predstavljanje znanja, automatsko dokazivanje teorema, procesiranje prirodnih jezika, deduktivne baze podataka, CAD sistemi, prepoznavanje oblika, robotika.

Relacioni upitni jezici SQL i QBE, detaljnije objašnjeni u odeljku 1.1., imaju implementacije na PROLOG-u. Upiti QBE jezika se prevode u PROLOG na sledeći način [Gray84]: svaki red obrasca relacije u QBE jeziku se prevodi u atom, čiji je simbol predikata ime relacije a argumenti su uzorci (promenljive) i konstante, koje respektivno odgovaraju redosledu navođenja vrednosti-uzoraka u redu tabele-obrasca. Prazne kolone se označavaju neimenovanim promenljivama u PROLOG-u (_). Slaganje između promenljivih u PROLOG-upitu i QBE elemenata-uzoraka je egzaktno, s obzirom da su i argumenti predikata PROLOG-klauzula i vrednosti u QBE tabeli egzistencijalno kvantifikovani nad domenima.

Primer 3.2. Date su sledeće šeme relacija:
SMER({ID_SMERA, NAZIV, BR_IND}, {ID_SMERA, BR_IND}), gde obeležje BR_IND ima semantiku: broj indeksa studenta sa najboljim uspehom na datom smeru,
i šema relacije:
STUDENT({BR_IND, IME, PREZIME, GOD_ROD},{BR_IND})

Upit "Prikazati ime i prezime studenta koji je prvi po uspehu na smeru INFORMATIKA" u QBE jeziku ima sledeću realizaciju:

STUDENT

BR_IND	IME	PREZIME	GOD_ROD
A	P	P	

SMER

ID_SMERA	NAZIV	BR_IND
	INFORMATIKA	A

U relaciji STUDENT izdvojiće se torka koja se za vrednost obeležja BR_IND (podvučeno A) poklapa sa vrednošću obeležja BR_IND torke relacije SMER gde je vrednost obeležja SMER="INFORMATIKA". Podvučeno slovo A je vrednost-uzorak i ta vrednost će se konkretizovati u momentu izvršenja upita. Simbol P u kolonama IME i PREZIME znači da će vrednosti tih obeležja biti prikazane korisniku.

Ovaj upit u PROLOG-u ima izgled:

odstampaj(X,Y):- *student*(A,X,Y,_),*smer*(_,'informatika',A).

3.2.2. Konačan neuspeh i problem negacije

PROLOG ne dopušta primenu negacije u glavi klauzula sa nepraznim telom. Negacija se tretira na specifičan način koji nije usaglašen sa tretmanom negacije u logičkim sistemima. Ovaj način je opisan u [Hot95]. U slučaju da je podcilj ili element podcilja negiran, podciljevi se generišu na sledeći način.

"Neka je cilj ?- C_1, C_2, ..., C_k, gde je C_1 oblika notC_1'. Nemogućnost zadovoljenja cilja C_1' (konačan neuspeh) znači da je u posmatranom skupu uslova cilj C_1' netačan, pa je cilj **not**C_1' zadovoljen kao tačan. Sa druge strane, mogućnost zadovoljenja cilja C_1' povlači nemogućnost zadovoljenja cilja notC_1' u skupu neprotivurečnih uslova. Dakle, pokušaj zadovoljenja cilja **?- notC** uspeva na osnovu toga što pokušaj zadovoljenja cilja **?- C** završava konačnim neuspehom."

Primer 3.3. Zadatak je postavljen u [Hot95] i komentarisano je njegovo rešenje na PROLOG-u str. 161-162.

Dat je program napisan na PROLOG-u:

```
visok(X1) :- not nizak(X1), covek(X1).
covek(marko).
covek(slavica).
nizak(mirko).
```

Na postavljeno pitanje "Da li je Slavica visoka?"

```
?- visok(slavica).
```

PROLOG-sistem daje odgovor:

```
Yes
```

Na postavljeno pitanje "Da li ima visokih ljudi?"

```
?- visok(Y1).
```

PROLOG-sistem daje odgovor:

```
No
```

Ovi odgovori su nekorektni i međusobno suprotstavljeni. Iz datih činjenica i pravila ne može se zaključiti da je Slavica niska ili visoka, nego samo da je Mirko nizak.

Na osnovu [Ber97] opisan je način na koji je PROLOG generisao ove odgovore.

Kod pokušaja zadovoljenja cilja:

```
?- visok(slavica)
```

generiše se novi cilj (unifikacija cilja sa glavom prve klauzule):

```
?- not nizak(slavica), covek(slavica)
```

Element prvog podcilja

?- nizak(slavica)

trpi konačan neuspeh, što znači da je zadovoljen prvi podcilj:

?- not nizak(slavica), covek(slavica)

Zatim se prelazi na drugi podcilj:

?- covek(slavica)

koji dovodi do generisanja praznog cilja sa pretposlednjom činjenicom iz programa, pa PROLOG-sistem daje odgovor Yes.

Kod pokušaja zadovoljenja cilja:

?- visok(Y1)

generiše se novi cilj (pri čemu se vrši unifikacija za Y1 i X1):

?- not nizak(X1), covek(X1)

Element prvog podcilja:

?- nizak(X1)

se može zadovoljiti pomoću poslednje činjenice iz programa (za mirko/X1), što znači da se prvi podcilj:

?- not nizak(X1)

ne može zadovoljiti pa posmatrani podcilj nema potomaka i predstavlja čvor neuspeha.

Ova nelogičnost se još više ispoljava ako se nekom od navedenih PROLOG-programa iz ovog primera postavi pitanje u vezi znanja koja nisu relevantna za dati problem. Na primer, ako se postavi pitanje:

?- voli(mirko,slavica)

PROLOG-sistem daje odgovor:

No

što se može tumačiti i kao "ne znam", nije izvodivo.

Međutim, ako se postavi pitanje:

?-not voli(mirko,slavica)

PROLOG-sistem daje odgovor:

Yes

mada relacija ***voli*** nije ni pomenuta u programu.

Ovakvo ponašanje sistema može se objasniti time što ako tvrđenje ***voli(mirko,slavica)*** ne sledi iz programa, onda ono nije relevantna istina u sistemu datih uslova, pa se njegova negacija usvaja kao istinit iskaz.

U [Ber97] je opisan "prologoliki" jezik, koji se zasniva na ADT sistemu. On tretira negaciju na logički korektan način, jer je izbegnuta koncepcija negacije kao konačnog neuspeha.

3.3. DATALOG i CWA-princip

3.3.1. Osnovne karakteristike DATALOG-a

U post-relacionim softverima za rukovanje bazama podataka: POSTGRES-u [Ma93] i ALGRES-u [CGT89] implementiran je DATALOG. DATALOG je paradigma logičkog programiranja i nastao je kao rezultat pokušaja spajanja logičkog programiranja sa bazama podataka.

Cilj razvoja DATALOG-a je bio razviti verziju PROLOG-a, ali tako da se omogući rukovanje velikom količinom podataka, koja se nalazi pohranjena u eksternoj memoriji.

DATALOG-program se sastoji od konačnog skupa činjenica i pravila. U formalizmu DATALOG-a činjenice i pravila se reprezentuju putem Hornovih klauzula:

$$L_o :- L_1,...,L_n$$

gde je literal L_i oblika $p_i(t_1,...,t_k)$, pri čemu je p_i predikatski simbol a t_j su termi. Term može biti konstanta ili varijabla.

Leva strana DATALOG klauzule zove se *glava* a desna strana *telo*. Telo klauzule može biti prazno. Klauzule sa praznim telom reprezentuju činjenice; klauzule sa najmanje jednim literalom u telu reprezentuju pravila.

U kontekstu logičkog programiranja se pretpostavlja da je svo znanje (činjenice i pravila), koje je relevantno za pojedinačnu aplikaciju, sadržano u pojedinačnom logičkom programu. DATALOG je, sa druge strane, razvijan za aplikacije koje koriste veliku količinu činjenica, pohranjenih u relacionoj bazi podataka. Iz tih razloga se u DATALOG-u razlikuju dva skupa klauzula:

- skup osnovnih činjenica, fizički pohranjenih u relacionoj bazi podataka. Ove klauzule se zovu ***Extensional Database*** (EDB)
- DATALOG-program ***P*** koji se zove ***Intensional Database*** (IDB) i definiše se kao skup ***pravila***. DATALOG-program može, takođe, da sadrži i činjenice.

Napomena: Pojmovi ***Extensional Database*** i ***Intensional Database*** se ne poklapaju sa pojmovima ***ekstenzija*** i ***intenzija*** u teoriji relacionog modela podataka. "Na nivou ekstenzije, koncepte relacionog modela podataka predstavljaju: domen obeležja, torka, relacija i pojava baze podataka, dok na nivou intenzije, osnovne koncepte predstavljaju: obeležje, šema relacije i šema baze podataka." ([ML96], str. 83).

Predikati, koji se pojavljuju u EDB i u programu ***P*** su podeljeni na dva disjunktna skupa:

- ***EDB-predikati***, koji se pojavljuju u bazi podataka, i
- ***IDB-predikati***, koji se pojavljuju u programu ***P***, ali ne i u bazi podataka (EDB).

U DATALOG-u postoji ograničenje da predikat glave svake klauzule programa ***P*** bude IDB-predikat. EDB-predikati mogu da se pojavljuju u programu ***P***, ali samo u telu klauzule.

Svaki EDB-predikat ***r*** odgovara tačno jednoj relaciji ***R*** baze podataka tako da je svaka činjenica $r(c_1,...,c_k)$ EDB-klauzule memorisana kao torka $(c_1,...,c_k)$ relacije ***R***.

IDB-predikati iz programa ***P*** mogu biti identifikovani sa relacijama, zvanim IDB-relacije ili *izvedene relacije*, koje su definisane putem programa ***P*** i EDB-predikata. IDB-relacije nisu pohranjene eksplicitno i one odgovaraju relacionim *pogledima*.

Primer 3.4. Data je relaciona baza podataka, koja se sastoji od dve šeme relacije:

OSOBA({ime},{ime})
RODITELJ({dete, roditelj}, {dete})

Pojave nad ovim šemama relacija su relacije:
relacija OSOBA

ime
Toma
Ana
Marko
Milica
Jovan

relacija RODITELJ

dete	roditelj
Marko	Toma
Milica	Jovan
Toma	Ana
Jovan	Ana

DATALOG-program P1 sastoji se od sledećih klauzula:

*r*1 : *ista_generacija*(X,X) :- *osoba*(X).
*r*2 : *ista_generacija*(X,Y) :- *roditelj*(X,X1), *ista_generacija*(X1,Y1), *roditelj*(Y,Y1).

Posledica pravila r1 je izvedena relacija ISTA_GENERACIJA, koja sadrži torke (osoba, osoba) za svaku osobu. Pravilo r2 je rekurzivno i njime se definiše da su dve osobe iste generacije rođaka kadgod imaju roditelje, koji su ista generacija rođaka. Sledeće torke pripadaju relaciji ISTA_GENERACIJA: (Toma, Jovan), (Marko,Milica).

Program P1 iz primera 3.4. je upit nad datom bazom podataka i on za rezultat daje relaciju ISTA_GENERACIJA. Baza podataka se posmatra kao vremenski promenljiva kolekcija informacija. Upit (program P1) je vremenski nepromenljivo preslikavanje, koje povezuje rezultat sa svakim mogućim stanjem baze podataka. Zbog toga je moguće formalno definisati semantiku DATALOG-programa P na sledeći način:

Definicija 3.11. DATALOG-program P je preslikavanje stanja baze podataka u rezultantno stanje.

Stanje baze podataka je kolekcija EDB-predikata a rezultantno stanje su IDB-predikati.

S obzirom da je DATALOG uprošćena verzija logičkog programiranja, koncept logičke posledice u DATALOG-u definisan je u terminima teorije modela.

Definicija 3.12. Činjenica F logički sledi iz skupa S klauzula, akko svaka interpertacija, koja zadovoljava svaku klauzulu C, takođe zadovoljava i F. Ako F sledi iz S to se označava sa $S \models F$.

Opšte pravilo zaključivanja, koje proizvodi nove DATALOG činjenice iz datih DATALOG činjenica i pravila se zove ***Elementarni Princip Izvođenja (Elementary Production Principle EPP)***. U daljem tekstu se prema [CGT89] opisuje ovo pravilo, navodi se definicija izvođenja i pojam stabla dokaza u DATALOG-u.

Posmatra se DATALOG-pravilo R oblika:

$$L_o :- L_1,. . ., L_n$$

i lista osnovnih činjenica $F_1,. . .,F_n$. (Literal, činjenica, pravilo ili klauzula koji ne sadrže promenljive zovu se *osnovnim* literalom, činjenicom, pravilom, klauzulom).

Definicija 3.13. Ako postoji zamena θ takva da za svako $1 \leq i \leq n$, $L_i\theta = F_i$, onda pravilo R iz činjenica $F_1,. . .,F_n$ *izvodi* u jednom koraku činjenicu $L_0\theta$.

Neka je S skup DATALOG klauzula. Osnovna činjenica F može biti *izvedena iz S*, u oznaci $S \vdash F$ akko je ili $F \in S$ ili se F može dobiti primenom pravila zaključivanja iz EPP konačan broj puta.

Definicija 3.14. Sekvenca primene Elementarnog Principa Izvođenja, koja se koristi da bi se izvela osnovna činjenica *F* iz skupa *S* zove se ***dokaz*** *F*-a iz *S*-a.

Svaki dokaz može biti reprezentovan putem ***stabla dokaza***, koje ima izvedenu činjenicu *F* u korenu.

DATALOG je sa sintaktičke tačke gledišta podskup PROLOG-a, s obzirom da se svaki skup DATALOG-klauzula može prevesti i izvršiti putem PROLOG-interpretera. Spajanje PROLOG-a sa eksternom bazom podataka je moguće, ali u tom postupku PROLOG-interpreter mora da pravi razliku između IDB-predikata i EDB-predikata. Kada se u toku izvršavanja PROLOG-programa odredi cilj, interpreter pokušava da pronađe odgovarajuću torku u bazi podataka. S obzirom na proceduralnu semantiku PROLOG-a, koja određuje redosled ciljeva i podciljeva, interakcija, koja se zahteva između interpretera i baze podataka u eksternoj memoriji je tipa *jedna-torka-u-jednom-momentu*. Ovaj metod pristupa masovnoj memoriji je veoma neefikasan u poređenju sa *skupovno-orijentisanim* metodama, koje se koriste u jezicima visokog nivoa.

DATALOG mora da obezbedi pristup većoj količini podataka u masovnoj memoriji. U nameri da se integrišu DATALOG i softver za rukovanje bazom podataka mora se uspostaviti veza između formalizma logičkog programiranja i nekog opšteg jezika baza podataka. U razvoju DATALOG-a odabrana je ***relaciona algebra*** kao upitni jezik.

Svaka klauzula DATALOG-programa se prevodi, putem sintaksno-upravljanog algoritma za prevođenje, u podskup relacija relacione algebre. Svaki IDB-predikat DATALOG-programa odgovara *promenljivoj relaciji* (relaciji koja može da sadrži i promenljive); svaki EDB-predikat DATALOG-programa odgovara *konstantnoj relaciji* (relaciji koja sadrži samo konstante).

Definicija 3.15. Data je DATALOG-klauzula [CGT89]:

$$C:\ p(\alpha_1,\ldots,\alpha_n)\ :-\ q_1(\beta_1,\ldots,\beta_k),\ldots,q_m(\beta_s,\ldots,\beta_h)$$

Prevođenje povezuje klauzulu C sa relacijama $Izraz(Q_1,\ldots,Q_m) \subseteq S$, (pri čemu je S skup relacija) tako da $Q_1,\ldots,Q_m$ odgovara predikatima p, $q_1,\ldots,q_m$, i važi konvencija da se atributi relacije imenuju brojem odgovarajućeg argumenta predikata sa kojim je relacija povezana.□□□

Primer 3.5. Posmatraju se program P1 i baza podataka iz **primera 3.4.**
DATALOG-program P1 sastoji se od sledećih klauzula:

*r*1 : *ista_generacija*(X,X) :- *osoba*(X).
*r*2 : *ista_generacija*(X,Y) :- *roditelj*(X,X1), *ista_generacija*(X1,Y1), *roditelj*(Y,Y1).

U bazi podataka se nalaze relacije OSOBA i RODITELJ.

Prevođenje ovog programa u relacije ima sledeći izgled:

$$\Pi_{1,5}((\text{RODITELJ} \underset{2=1}{\triangleright\triangleleft} \text{ISTA\_GENERACIJA}) \underset{4=2}{\triangleright\triangleleft} \text{RODITELJ}) \subseteq \text{ISTA\_GENERACIJA}$$

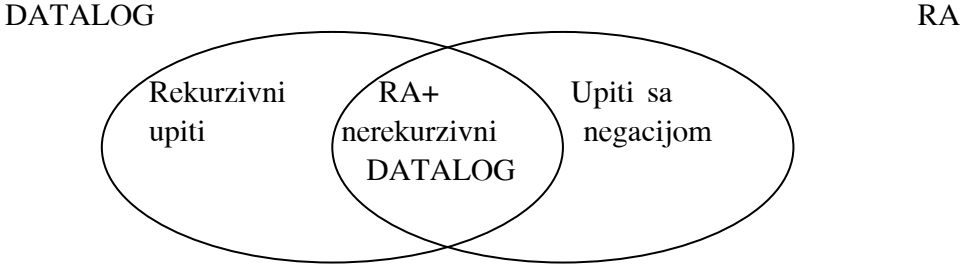
$$\Pi_{1,1}(\text{OSOBA}) \subseteq \text{ISTA\_GENERACIJA}$$

Pravilo prevođenja je da literali sa zajedničkim promenljivama određuju operaciju *spoja* ($\triangleright\triangleleft$) relacione algebre, dok literal u glavi klauzule određuje operaciju *projekcije* (Π).

Sistem, koji nastaje prevođenjem, uključuje sve klasične operacije relacione algebre, izuzev operacije *razlike* i taj sistem se označava kao *pozitivna relaciona algebra* RA+.

Operacija *razlike* nije implementirana u ***čistom*** DATALOG-u. Na primer, za dve binarne relacije R i S nije definisano DATALOG-pravilo za operaciju razlike R - S. Ova operacija je implementirana ekstenzijom čistog DATALOG-a sa logičkom negacijom ($\neg$).

Na slici 3.4. je predstavljena ekspresivna moć čistog DATALOG-a.



Slika 3.4.

3.3.2. Zatvoreni svet i CWA-princip

U čistom DATALOG-u nije implementirana logička negacija ($\neg$). Zbog toga je nemoguće izraziti pravila koja sadrže negativne pretpostavke. Na primer, "Ako je X student i X *nije* diplomirani student, onda je X nediplomirani student." U relacionoj algebri ovakva pravila se realizuju putem operacije *razlike* relacione algebre.

Primer 3.6. Date su relacije:
STUDENT, čija je semantika spisak imena studenata.
DIPL_STUD, čija je semantika spisak imena diplomiranih studenata.

Tada se relacija NEDIPL_STUD (spisak imena nediplomiranih studenata) može dobiti primenom operacije razlike nad relacijama STUDENT i DIPL_STUD:

NEDIPL_STUD = STUDENT - DIPL_STUD

U nameri da se omogući izražavanje negativnih pretpostavki, čisti DATALOG se proširuje tako što se dopušta prisustvo *negativnih literala* (literala kojima prethodni simbol negacije ¬) u telu pravila. Takav DATALOG se označava sa ***DATALOG¬***.

Primer 3.7. Date su relacije STUDENT i DIPL_STUD iz **primera 3.6**. Neka predikatski simboli *student* i *dipl_stud* reprezentuju relacije STUDENT i DIPL_STUDENT respektivno. Ako se relacija NEDIPL_STUD prestavi predikatskim simbolom *nedipl_stud* onda se implementacija pravila: "Ako je X student i X *nije* diplomirani student, onda je X nediplomirani student." realizuje sledećom klauzulom DATALOG¬-a:

nedipl_stud(X) :- *student*(X), ¬ *dipl_stud*(X).

DATALOG radi po principu ***pretpostavke zatvorenog sveta*** (**Closed World Assumption**) odnosno po **CWA-principu.** Ovaj princip omogućava izvođenje negativnih činjenica iz skupa klauzula DATALOG-a. Drugim rečima, za svaku činjenicu, koja se ne nalazi u bazi podataka, na osnovu ovog principa, proglašava se da je netačna (negacija te činjenice je istina).

Definicija CWA-principa se nalazi u [Ullm85], [Ullm88] i [CGT89] i ona glasi:

Definicija 3.16. (**CWA-princip**) Ako činjenica ne proizilazi logički iz skupa DATALOG klauzula, onda se zaključuje da je negacija te činjenice istina.

Ova pretpostavka je karakteristična za koncept upitnih jezika baza podataka. Upitni jezici (SQL, QBE) na osnovu zadatog kriterijuma i strategija pretrage vrše pretraživanje u bazi podataka. Ako je proces traženja (pretraživanja) u bazi podataka završen uspešno, sledi odgovarajuća poruka ("Podatak je pronađen", ili se sam podatak prikaže korisniku), a ako je proces završio neuspehom odgovor je "Nema tog podatka u bazi podataka".

Dakle, ***princip zatvorenog sveta*** važi kao princip na kome se temelji rad upitnih jezika baza podataka, pa samim tim i jedne verzije jezika logičkog programiranja za baze podataka - DATALOG-a.

Primer 3.8. Dat je skup DATALOG-klauzula: Sc={*dosadan*(šah):- ¬ *interesantan*(šah)}. Ovaj skup se "*evaluira*" (termin DATALOG-a i znači: traži se rešenje, koje zadovoljava skup činjenica i pravila) na sledeći način: pre pokušaja evaulacije predikata *dosadan*, evaluira se predikat *interesantan*, koji se pojavljuje negativno u telu pravila. S obzirom da

nema pravila i činjenica u skupu Sc, koje bi omogućile izvođenje bilo koje činjenice za *interesantan*(α), skup pozitivnih odgovora za ovaj predikat je prazan. To znači, da predikat *interesantan*(šah) ne može biti izveden, pa se primenom CWA-principa "lokalno" na

predikat *interesantan*, izvodi ¬ *interesantan*(šah), odnosno negacija te činjenice se proglašava za istinu. Za rezultat se dobija *dosadan*(šah).

Razvoj DATALOG-a je išao u pravcu razvoja posebne tehnike: ***tehnike raslojavanja***. Suština ove tehnike je u načinu tretiranja literala u DATALOG-klauzulama tokom izvođenja dokaza (*evaluacije predikata* u terminima DATALOG-a [CGT89]) i ona se sastoji u sledećem: kada se izvodi (evaluira) pravilo sa jednim ili više negativnih literala u telu pravila, prvo se pokušavaju zadovoljiti (evaluirati) predikati, koji odgovaraju tim negativnim literalima. Zatim se CWA-princip primenjuje "lokalno" na te predikate.

Znači, zahteva se da se pre evaluacije predikata u glavi pravila izvrši kompletna evaluacija svih predikata koji se pojavljuju negativno u telu pravila ili u telima svih onih pravila, koji su neophodni za evaluaciju tih negativnih predikata. Ako DATALOG-program zadovoljava ovo pravilo onda se on zove ***višeslojni DATALOG***.

Na taj način se DATALOG-program P deli na disjunktne skupove klauzula:

$$P = P^1 \cup P^2 \cup P^i \cup \ldots \cup P^n$$

i svaki od tih skupova zove se ***nivo (sloj)***.

Pravilo za raslojavanje DATALOG-programa glasi:
Dat je raslojeni program P sa raslojavanjem $P^1 \ldots P^n$ i on se evaluira nad bazom podataka EDB *E*. Prvo se P^1 evaluira primenom CWA lokalno nad EDB, odnosno pod pretpostavkom: $\neg p(c_1,\ldots,c_k)$ za svaki *k*-ti EDB-predikat *p* i konstante $c_1,\ldots,c_k$ pri čemu $p(c_1,\ldots,c_k) \notin E$. Ostali slojevi se evaluiraju u rastućem redosledu. Tokom evaluacije svakog nivoa P^i, koriste se rezultati prethodnih izračunavanja i CWA se "lokalno" primenjuje za sve EDB-predikate.

Primer 3.9. Dat je program Ps pri čemu je sa *d* označen EDB-predikat:

r1: $p(X,Y) :- \neg q(X,Y), s(X,Y).$
r2: $q(X,Y) :- q(X,Z), q(Z,Y).$
r3: $q(X,Y) :- d(X,Y), \neg r(X,Y).$
r4: $r(X,Y) :- d(Y,X).$
r5: $s(X,Y) :- q(X,Z), q(Y,T), X \neq Y$

Raslojavanje programa Ps je:

$Ps^1 = \{ r4 \}$
$Ps^2 = \{ r2, r3, r5 \}$
$Ps^3 = \{ r1 \}.$

Program Ps je evaluiran nad bazom podataka EDB $E = \{ d(a,b), d(b,c), d(e,e) \}$.
Evaluacija prvog nivoa proizvodi nove činjenice:

$r(b,a), r(c,b), r(e,e)$

Izračunavanje drugog nivoa proizvodi nove dodatne činjenice:

q(a,b), *q*(b,c), *q*(a,c), *s*(a,b), *s*(b,a).

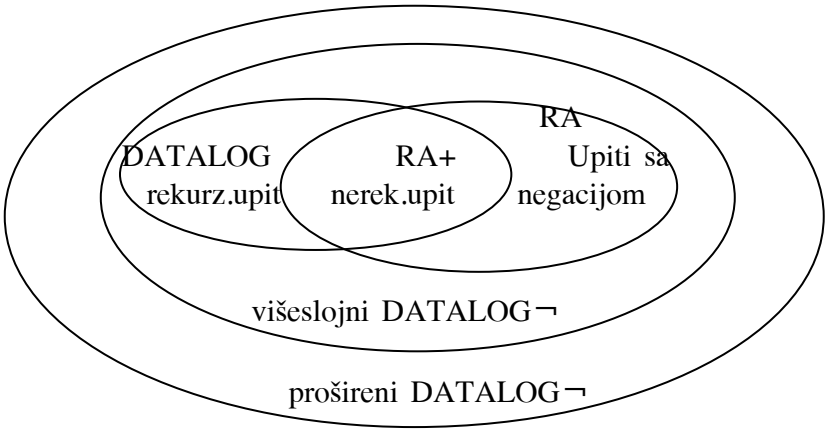
Na kraju, evaluacijom trećeg nivoa dobija se:

p(b,a).

Osnovni princip izvođenja u DATALOG-u je CWA-princip: u slučaju da činjenica ne sledi iz skupa klauzula i činjenica usvaja se negacija te činjenice za istinu. Tehnikom raslojavanja omogućen je samo efikasniji način zadovoljenja cilja DATALOG-programa.

Osim ***višeslojnog DATALOG-a*** razvijen je i ***prošireni DATALOG***. U ovoj verziji DATALOG-a evaluacija programa P nad bazom podataka E se izvodi iterativno tako da se sva pravila P izvršavaju *paralelno* u svakom koraku. Na taj način je realizovana još jedna tehnika za efikasniju evaluaciju DATALOG-programa i sa stanovišta osovne ideje ovog rada nema posebno bitne karakteristike (detaljnije o tehnici u [CGT89]).

Hijerarhija izražajne moći različitih verzija DATALOG-a data je na slici 3.5., gde RA označava relacionu algebru.



Slika 3.5.

4. POVEZIVANJE KONCEPATA OTVORENOG I ZATVORENOG SVETA U JEDINSTVEN SISTEM BASELOG

4.1. Osnovna hipoteza i motivacija za istraživanje

Kao što je objašnjeno u odeljku 3.3.2., jezik logičkog programiranja u bazama podataka DATALOG radi po principu CWA odnosno usvajanjem pretpostavke ***zatvorenog sveta.*** Ova pretpostavka je prisutna u bazama podataka koje se koriste u *komercijalne svrhe* (baze podataka informacionih sistema u preduzećima, bankama, zdravstvenim ustanovama, avio kompanijama, državnim institucijama itd.). Semantika ovih baza podataka je takva da je potrebno samo pronaći odgovarajući podatak u bazi a onda nad njim vršiti ili ažuriranje ili samo prikaz podataka korisniku.

Za *baze znanja* je karakteristična i pretpostavka ***otvorenog sveta***, koja je predodređena njihovom semantikom. U bazi podataka koja se koristi u obrazovnom računarskom softveru nalaze se informacije o nekoj predmetnoj oblasti (geografija, matematika, biologija, istorija) i one su prezentovane putem koncepata odabranog modela podataka (mreže, stabla, tabele, objekti). S obzirom da svaka baza sadrži uvek ograničen segment znanja iz određene oblasti i da je nemoguće predstaviti ***ukupno relevantno znanje*** u bazi podataka, postoji mogućnost dobijanja netačnih odgovora na postavljene upite od strane učenika ako se usvoji koncept baze podataka kao **zatvorenog sveta**. Zbog toga je čist koncept **zatvorenog sveta** neprimenljiv za baze podataka koje se koriste u obrazovnom računarskom softveru.

Primer 4.1. Neka se u bazi podataka nalaze sledeće informacije:

- Prave A i B su paralelne
- Prave B i C su paralelne

I neka u bazi podataka ne postoji znanje o tranzitivnosti paralelnosti pravih.
Na učenikovo pitanje "*Da li su prave A i C paralelne?*" u slučaju da se baza podataka obrazovnog softvera posmatra kao zatvoren svet, sistem će generisati odričan odgovor "**nisu**". Taj odgovor je netačan i dezinformiše učenika, što se ne sme dopustiti. U slučaju kada bi se ista baza posmatrala kao otvoren svet, sistem bi odgovorio "**nije mi poznato**", odnosno, "**neizvesno**", što je sa stanovišta informisanja učenika prihvatljivo. □

Sa druge strane, rezolucijski sistemi poseduju teorijsku univerzalnost, odnosno teorijsku snagu za izvođenje svakog zaključka kad god je on logička posledica datih pretpostavki. Zato takvi sistemi u potpunosti automatizuju proces logičkog zaključivanja (rezonovanja) i daju ispravne odgovore u odnosu na zadate polazne pretpostavke. Upravo to takvim sistemima daje teorijsku univerzalnost. Univerzalni rezolucijski sistemi sa teorijskog aspekta u potpunosti podržavaju i rad sa bazama podataka ali ispoljavaju praktičan nedostatak. On se ogleda u tome što je zbog nastojanja da se dobije semantički očekivani odgovor neophodno dati kompletan opis prostora u kome se traži rešenje.

U domenu baza podataka ovo se, na primer, manifestuje kroz neophodnost zadavanja sledeće aksiome:

$$t \neq t_1 \wedge t \neq t_2 \wedge \ldots \wedge t \neq t_n \Rightarrow \neg P(t) \qquad (1)$$

gde sa $t_1,\ldots,t_n$ označavamo torke relacione baze podataka a sa P(t) pripadnost torke relaciji baze podataka.

Kao što se vidi, već i za mali broj torki u bazi ova aksioma ima veliku dužinu pa se odustalo od ove teorijske mogućnosti u praktičnim primenama.

I u DATALOG-u i u PROLOG-u je učinjen pokušaj rešavanja ovog nedostatka i to na specifične načine kao što je objašnjeno u odeljku 3.2. i 3.3. U PROLOG-u je to strategija konačnog neuspeha a u DATALOG-u princip CWA. Međutim, ni jedno od ovih rešenja ne može u potpunosti da zadovolji potrebe obrazovanja, što ilustruje sledeći primer.

Primer 4.2. Neka se u bazi podataka nalaze samo sledeći elementi:

TROUGAO(zbir_unutrasnjih_uglova, 180)
KVADRAT(zbir_unutrasnjih_uglova, 360)
PRAVOUGAONIK(povrsina, ab)
KRUG(obim, $2r\pi$)

U odnosu na moguće upite učenika (korisnika) razlikuju se sledeće opcije:

a) odgovor na upit je izvodiv iz baze,
b) odgovor na upit nije izvodiv iz baze,

pri čemu se u slučaju b) razlikuju:
b1) odgovor treba da bude potvrdan,
b2) odgovor treba da bude odričan.

U slučaju a) kada je odgovor izvodiv iz baze, on će biti nađen i prezentovan korisniku bilo da se koristi PROLOG, DATALOG ili ADT sistem.

Tako na upit **?- trougao(zbir_unutrasnjih_uglova,180)** odgovor svakog od ovih sistema je potvrdan, što je u skladu sa očekivanjem.

Specifičnosti se ispoljavaju u slučaju b).

Slučaj b1) Korisnik može da postavi sledeći upit:

?- krug(povrsina, $r^2\pi$)

Odgovori sistema su:

- PROLOG: **no**, što se može shvatiti kao **neizvesno**
- DATALOG: **NE**, zbog CWA-principa
- ADT: **neizvesno**

U ovom slučaju DATALOG daje nedopustiv i pedagoški neprihvatljiv odgovor, dok PROLOG i ADT daju neodređen odgovor, koji je pedagoški prihvatljiv.

Slučaj b2) Korisnik može da postavi sledeći upit:

?- trougao(zbir_unutrasnjih_uglova, 360)

Odgovori sistema su:

- PROLOG: **no**,
- DATALOG: **NE**,
- ADT: **neizvesno**

U ovom slučaju odgovor DATALOG-a je korektan i određen, dok PROLOG i ADT daju neodređen odgovor.

Isti odgovori dobijaju se i kad upit nije u vezi sa elementima baze, kao na primer:

?- krug(povrsina, $2r\pi$)

ili

?- elipsa(obim, $r^2\pi$)

Primećuje se da u pedagoškom smislu DATALOG zbog slučaja b1) ne zadovoljava, dok PROLOG i ADT daju prihvatljive, ali neodređene odgovore.

U slučaju b2) DATALOG daje korektan i precizan odgovor, dok PROLOG i ADT daju nedovoljno precizne odgovore.

Sa pedagoškog aspekta poželjno je maksimalno smanjiti neodređenost odgovora sistema i nužno je eliminisati nedopustive odgovore. Drugim rečima, treba sačuvati određenost prisutnu u DATALOG-u za slučaj b2) i eliminisati nedopustiv odgovor iz slučaja b1). To se može postići pomoću uvođenja liste CWA-predikata na sledeći način.

Neka je predikat TROUGAO projektant baze deklarisao kao CWA-predikat. To znači, da će samo za njega važiti princip CWA, dok za sve ostale neće.

Sada sistem BASELOG, koji koristi takvu listu deklarisanih CWA-predikata, daje sledeće odgovore:

- Na upit iz slučaja b1) **?- krug(povrsina, $r^2\pi$)** daje odgovor **NEIZVESNO**, odnosno radi u režimu ***otvorenog sveta*** kao ADT sistem, jer predikat **KRUG** nije u listi.
- Na upit iz slučaja b2) **?- trougao(zbir_unutrasnjih_uglova, 360)** daje odgovor **NE**, odnosno radi u režimu ***zatvorenog sveta*** kao DATALOG, jer je predikat **TROUGAO** u listi CWA-predikata.

Projektant baze podataka u takvom BASELOG-sistemu, izborom liste CWA-predikata, upravlja ponašanjem sistema i omogućuje dobijanje **preciznih i korektnih** odgovora u situacijama u kojima PROLOG i ADT daju neodređene odgovore i eliminiše se mogućnost nedopustivih odgovora.

Pri formiranju liste CWA-predikata treba pažljivo analizirati sadržaj baze podataka i moguće upite korisnika. Tako, ne bi bilo dobro predikat KRUG deklarisati kao CWA-predikat, jer u bazi podataka nisu sadržane sve relevantne činjenice u vezi sa krugom. Takođe, i predikat TROUGAO se sme uzeti za CWA-predikat samo ako se unapred zna da korisnik neće postavljati upite, koji se ne odnose na zbir unutrašnjih uglova izražen u stepenima. Ograničenje samo na stepene može se postići uvođenjem posebnog predikata **TROUGAO_Z_U_STEP(unutrasnji,180)** sa semantikom: **Z**bir **U**glova u **S**tepenima. Sada se predikat može sa sigurnošću proglasiti za CWA-predikat. Uopšte, za CWA-predikat se može deklarisati samo onaj predikat iz baze za koga želimo da sistem radi u režimu zatvorenog sveta. To je naročito pogodno kad je reč o predikatima koji se javljaju u testiranju, kao rešenja, ili o definicijama pojmova.

Na primer, ako je rešenje prvog zadatka u testu, na primer, 325, odnosno **REŠENJE_ZAD1**(325), onda se predikat **REŠENJE_ZAD1** mora proglasiti za CWA-predikat.

Međutim, implementacijom sistema BASELOG, zasnovanog na listi CWA-predikata, predlaže se fleksibilniji koncept, koji eliminiše neodređenost i nedopustive odgovore prisutne u PROLOG-u, DATALOG-u i ADT sistemu.

Takav sistem sve predikate koji su u listi CWA-predikata tretira u zatvorenom svetu, odnosno radi kao DATALOG, dok sve ostale predikate tretira u otvorenom svetu, odnosno, radi kao ADT. Pri tome, oslobođen je nedostatka PROLOG-a u odnosu na tretman negacije i koncepciju konačnog neuspeha.

Na osnovu opisanih saznanja mogu se odrediti sledeće hipoteze ovog istraživanja.

OSNOVNA HIPOTEZA

Moguće je povezati koncepte otvorenog i zatvorenog sveta u jedinstven sistem koji obezbeđuje rukovanje bazom u režimu otvorenog, zatvorenog ili delimično otvorenog - zatvorenog sveta.

POMOĆNE HIPOTEZE

- Osim teorijskog koncepta takvog fleksibilnog sistema, može se izgraditi i implementirati konkretan programski sistem BASELOG, koji predstavlja proširenje postojećeg ADT sistema zasnovanog na uređenoj OL rezoluciji sa markiranim literalima, bez potrebe za zadavanje aksioma kojima se kompletira prostor traženja rešenja.

- Takav programski sistem (BASELOG) omogućuje fleksibilno rukovanje realno postojećim bazama podataka u željenom režimu rada, ne samo za klasični tip baza podataka nego i baza znanja, posebno za one baze znanja, koje se koriste u projektovanju obrazovnog računarskog softvera.

- Putem posebnih interfejsa može se obezbediti uključivanje realne baze u režim rada BASELOG-sistema (učitavanje cele baze u operativnu memoriju, ili zadržavanje baze na eksternom medijumu uz učitavanje samo aktuelnog segmenta).

Cilj je izgraditi sistem koji neće raditi samo na osnovu ***jednog*** principa, nego će imati mogućnost prilagođavanja režima u toku rada. Ova mogućnost je realizovana definisanjem liste CWA-predikata. To znači da za predikate, koji su definisani u **listi CWA** sistem radi u režimu zatvorenog sveta, a za sve ostale predikate radi u režimu otvorenog sveta. Umesto jednog generalnog principa, na kome se temelji celokupan rad sistema, ovde se definisanjem liste CWA-predikata određuje način rada u konkretnoj situaciji. Znači, cilj je omogućiti ***fleksibilnost*** rada sistema u zavisnosti od tipa predikata.

4.2. CWA-predikati i CWA-pravilo

Osnovu za razvoj BASELOG-sistema čine sledeće komponente:

- **lista CWA-predikata**, koja je deo programa,
- **CWA-pravilo**,
- **CWA-kontroler** kojim je proširen rezolucijski ADT sistem.

Univerzalna rezolucijska procedura je ***proširena*** dodavanjem CWA-kontrolera, koji obezbeđuje definisanje režima rada u zatvorenom svetu i to ili za sve, ili samo za neke

ili ni za jedan predikat iz programa. Ceo BASELOG-sistem je ekstenzija rezolucijske metode konceptima otvorenog i zatvorenog sveta. Putem CWA-kontrolera obezbeđuje se doziranje stepena otvorenosti / zatvorenosti sveta za predikate iz programa.

Definicija 4.1. Neka je $R(w_1,...,w_m)$ literal, gde je R naziv predikata a $w_1,...,w_m$ su termi. R je **CWA-predikat** akko ga sistem obrađuje u režimu zatvorenog sveta.

U suprotnom R nije CWA-predikat i sistem će ga obraditi u režimu otvorenog sveta.
CWA-pravilo se formuliše na sledeći način:

CWA-PRAVILO

- *Neka je D sastavak oblika $L_1 \vee L_2 \vee ... \vee L_p$, gde je L_i, $1 \leq i \leq n$, literal bez negacije oblika $R(w_1,...,w_m)$ i predikat R je deklarisan kao CWA-predikat. Ako se $R(w_1,...,w_m)$ ne može unificirati ni sa jednim elementom baze, onda se $R(w_1,...,w_m)$ briše iz sastavka D.*
  U suprotnom, ako postoji unifikator za R i neki element baze, onda se sastavak D ne menja odnosno nema brisanja. Kažemo da se CWA-pravilo ne može primeniti na R i D. □

Dokaz korektnosti CWA-pravila:

Za dokaz korektnosti CWA-pravila dovoljno je dokazati da kad god se CWA-pravilo može primeniti ono generiše rezolventu koja bi se u čisto rezolucijskoj proceduri generisala uz pomoć dodatne aksiome za zatvaranje sveta u odnosu na aktuelni predikat.

Neka je na nivou l rezolucijom generisana rezolventa:

$$D: \quad L_1 \vee L_2 \vee ... \vee L_p \tag{2}$$

gde je literal L_i, $1 \leq i \leq n$, oblika $R(w_1,...,w_m)$, pri čemu je R deklarisan kao CWA-predikat. Sledi, da sistem treba da obradi R u režimu zatvorenog sveta. U sistemu koji koristi samo rezoluciju, neophodna je dodatna aksioma kojom se obezbeđuje tretman predikata R u zatvorenom svetu.

Moguća su dva slučaja:
a) predikat R se ne pojavljuje u bazi

b) predikat R se pojavljuje u bazi za torke $t_1,t_2,...,t_k$, $k \leq n$ (n je broj svih torki u bazi).

U slučaju a) dodatna aksioma je: $\neg R(t)$, za svako t. (Ax.1)

U slučaju b) dodatna aksioma je:

$$t \neq t_1 \wedge t \neq t_2 \wedge \ldots \wedge t \neq t_k \Rightarrow \neg R(t) \qquad \text{(Ax.1')}$$

u obliku sastavaka:

$$t = t_1 \vee t = t_2 \vee \ldots \vee t = t_k \vee \neg R(t) \qquad \text{(Ax. 1'')}$$

Slučaj a)
Rezolucijom iz D i Ax.1 izvodi se rezolventa:

$$(D \setminus \{L_i\})_\theta: \qquad L_{1\theta} \vee \ldots \vee L_{i-1\theta} \vee L_{i+1\theta} \vee \ldots \vee L_{p\theta}$$

gde je θ unifikator za $\boldsymbol{t}$ i $(w_1,...,w_m)$. U stvari, $\boldsymbol{t}$ je oblika $(x_1,...,x_m)$, gde su x_i, i=1,...,m promenljive, pa je $\theta = \{w_1/x_1,...,w_m/x_m\}$. Kako Ax.1 i D ne sadrže zajedničke promenljive (to se uvek može postići preoznačavanjem, videti u [Hot95]) sledi da zamena θ ne menja literale $L_1,...,L_p$. Zato je rezolventa $(D \setminus \{L_i\})_\theta$: $L_1 \vee \ldots \vee L_{i-1} \vee \vee L_{i+1} \vee \ldots \vee L_p$.

Upravo to se dobija i primenom CWA-pravila na D. Zaista, u ovom slučaju ni za jedan elemenat baze ne postoji unifikator za $R(w_1,...,w_m)$ (jer u bazi nema predikata R).

Time je dokaz slučaja a) završen.

Slučaj b)
Rezolucijom iz D i Ax.1'' izvodi se rezolventa:

$$L_{1\theta} \vee \ldots \vee L_{i-1\theta} \vee L_{i+1\theta} \vee \ldots \vee L_{p\theta} \vee t_\theta = t_{1\theta} \vee \ldots \vee t_\theta = t_{k\theta}$$

gde je θ unifikator za $\boldsymbol{t}$ i $(w_1,...,w_m)$ pri čemu ta zamena deluje samo na ostatak Ax.1'' i ne deluje na torke $t_1,...,t_k$ (one sadrže konstante), pa je rezolventa:

$$L_1 \vee \ldots \vee L_{i-1} \vee L_{i+1} \vee \ldots \vee L_p \vee t_\theta = t_1 \vee \ldots \vee t_\theta = t_k$$

Sada su moguća dva slučaja: b1) i b2).

Slučaj b1) Ni jedna od jednakosti ne važi, odnosno

$$t_\theta \neq t_1,...,t_\theta \neq t_k,$$

pa se izvodi rezolventa:

$$L_1 \vee \ldots \vee L_{i-1} \vee L_{i+1} \vee \ldots \vee L_p$$

Upravo to se dobija i primenom CWA-pravila na D, jer u ovom slučaju t_θ je $(w_1,...,w_m)$ i ne postoji unifikator ni sa jednim elementom $t_1,...,t_k$ iz baze.

Slučaj b2) Neka od jednakosti je zadovoljena, odnosno važi $t_\theta=t_j$, ($j \in \{1,2,...,n\}$). Tada u bazi postoji element $R(w_1,...,w_m) \equiv R(t_\theta) \equiv R(t_j)$, pa se CWA-pravilo u ovom slučaju ne može primeniti.
Time je dokaz kompletiran.

Sada se upotreba CWA-pravila može ovako rezimirati:

Neka je sastavak

$$D: L_1 \vee L_2 \vee ... \vee L_{p-1} \vee L_p$$

koji sadrži CWA-predikat u literalu L_i, ($1 \leq i \leq p$)

Slučaj 1⁰ **Pokušaj unifikacije literala L_i, ($1 \leq i \leq p$), sa nekim elementom iz baze je uspeo, onda sastavak D ostaje nepromenjen, odnosno brisanje iz CWA-pravila se ne primenjuje.**

Slučaj 2⁰ **Pokušaj unifikacije literala, L_i, ($1 \leq i \leq p$), sa svakim elementom iz baze nije uspeo. Onda CWA-pravilo briše literal L_i iz sastavka D.**

Primer 4.3. U ovom primeru biće ilustrovan slučaj b) - izvođenje zaključka u slučaju dodatne aksiome Ax.1''. Zadatak je definisan u [LHRB95] i odnosi se na problem zadovoljenja generalizovanih zavisnosti podataka (E i T zavisnosti) metodom automatskog rezonovanja - upotrebom ADT sistema. Na ovom mestu uzeće se kao primer samo dokaz zadovoljenja T-zavisnosti. T-zavisnost je formulisana u [ML96].

Za dokaz da T-zavisnost $<\tau, \ell>$, gde je $\tau = \{(x_1,y_1,z_1,w_1), (x_1,y_2,z_2,w_2)\}$ i $\ell = (x_1,y_1,z_2,w_2)$, u relaciji r nije zadovoljena pomoću putem ADT sistema definisan je sledeći polazni skup sastavaka S:

1. $\neg P(x_1,y_1,z_1,w_1) \vee \neg P(x_1,y_2,z_2,w_2) \vee P(x_1,y_1,z_2,w_2)$ (teorema o T - zavisnosti)

Torke relacije r, kao elementi baze podataka, predstavljene su putem predikata P:
2. $P(a_1,b_1,c_1,d_1)$
3. $P(a_2,b_1,c_2,d_2)$
4. $P(a_1,b_2,c_1,d_1)$
5. $P(a_1,b_2,c_2,d_2)$

Dugačka aksioma kojom se zadaje kompletan opis prostora pretrage je oblika:

6. $(x,y,z,w) = (a_1,b_1,c_1,d_1) \vee (x,y,z,w) = (a_2,b_1,c_2,d_2) \vee (x,y,z,w) = (a_1,b_2,c_1,d_1) \vee$
   $\vee (x,y,z,w) = (a_1,b_2,c_2,d_2) \vee \neg P(x,y,z,w)$

U implikativnoj formi aksiome 1. i 6. glase:

1. $(P(x_1,y_1,z_1,w_1) \wedge P(x_1,y_2,z_2,w_2)) \Rightarrow P(x_1,y_1,z_2,w_2)$ (uslov T - zavisnosti)
2. $((x,y,z,w) \neq (a_1,b_1,c_1,d_1) \wedge (x,y,z,w) \neq (a_2,b_1,c_2,d_2) \wedge (x,y,z,w) \neq (a_1,b_2,c_1,d_1) \wedge \wedge (x,y,z,w) \neq (a_1,b_2,c_2,d_2)) \Rightarrow \neg P(x,y,z,w)$

Na osnovu ovako definisanog skupa uslova sistem ADT je generisao prazan sastavak, što znači da relacija ***r***, koja je predstavljena putem datih torki, ne zadovoljava T-zavisnost, što je i trebalo dokazati.

Izvođenje dokaza uz prisustvo CWA-kontrolera

Sada će se pokazati kako se ovaj isti primer realizuje upotrebom BASELOG-sistema, ali putem liste CWA-predikata, kojom se eliminiše pisanje dugačke aksiome.

Lista CWA-predikata sadrži samo jedan predikat 'P', i za njega BASELOG-sistem radi u režimu zatvorenog sveta. Aksiome 1.-5. su iste, dok dugačka aksioma 6. nije potrebna.

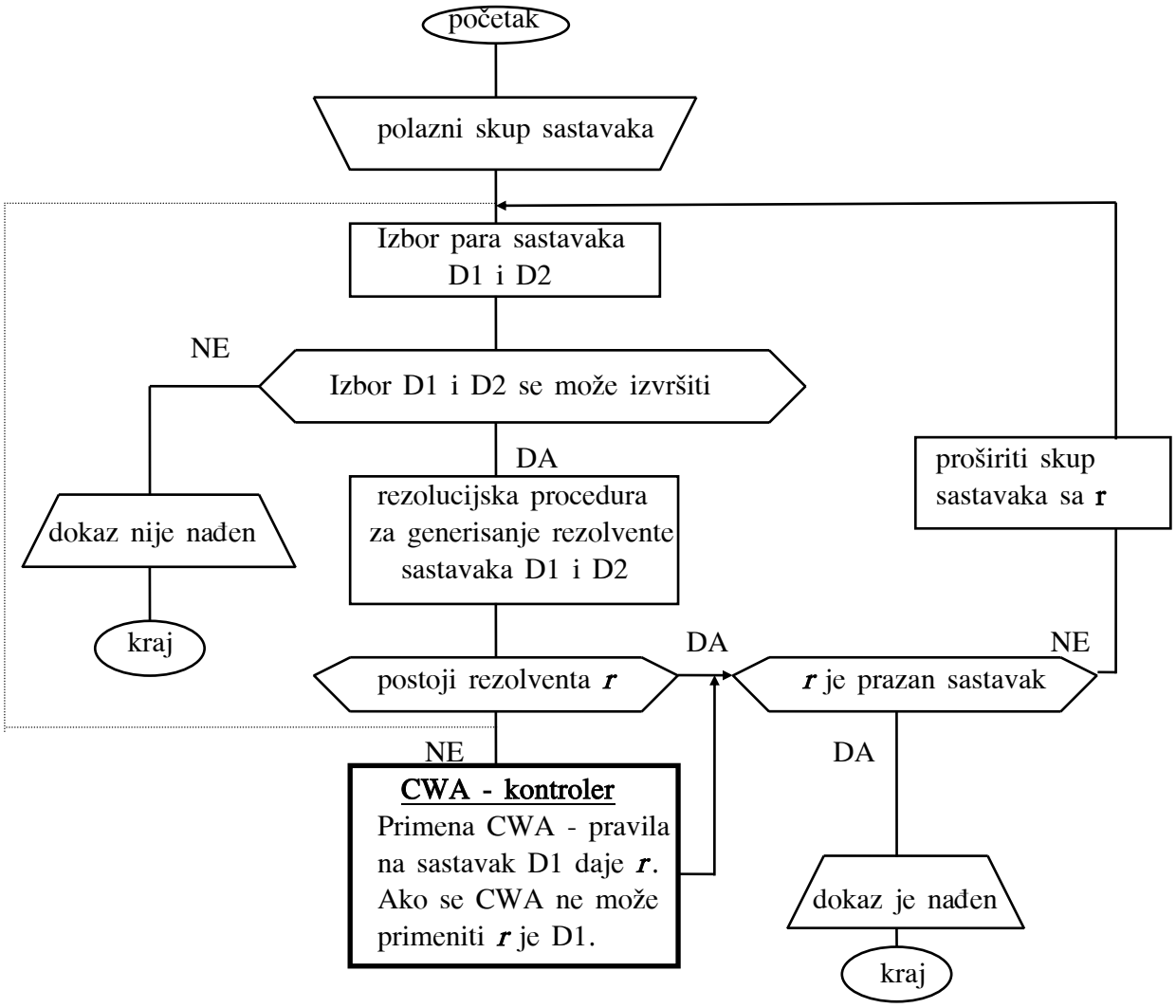
BASELOG-sistem generiše prazan sastavak, što znači da daje isti rezultat kao ADT sistem uz dugačku aksiomu.

4.3. Princip ugrađivanja CWA-pravila u rezolucijske procedure

CWA-pravilo koje je ovde uvedeno omogućuje da se dodatne aksiome za predikate, koje treba obraditi u režimu zatvorenog sveta eliminišu iz polaznog skupa aksioma, odnosno da se uopšte ne pišu. To oslobađa od rada sa dugačkim aksiomama kojih je u praksi često i nemoguće efektivno zapisati. Ugrađivanje CWA-pravila u rezolucijske procedure se može ostvariti na bazi sledećeg principa:

- **CWA-pravilo se primenjuje na dati sastavak (rezolventu) samo ako se na taj sastavak više ne može primeniti pravilo rezolucije, a svi uslovi za primenu CWA-pravila su ispunjeni.**

Ovaj princip ne zavisi od konkretne forme rezolucije, ni od konkretne strategije pretraživanja, pa se može primeniti na svaku od njih. Opšta blok šema za rezoluciju modifikuje se bez narušavanja osnovne strukture dodavanjem modula CWA-kontrolera (Slika 4.1.) Isprekidana linija na slici 4.1. je navedena radi komparacije sa sistemom koji se zasniva samo na rezoluciji bez CWA-kontrolera.



Slika 4.1.

Razlog za uvođenje ovog principa, kojim se prednost daje rezoluciji, je što u programu koji koristi bazu podataka, čiji se elementi shvataju kao aksiome, mogu postojati i posebne tzv. *sopstvene aksiome*. Sopstvenim aksiomama se mogu uvesti i predikati koji ne postoje u bazi a koji se određuju i povezuju sa predikatima iz baze.

Primer 4.4. Neka elementi baze sadrže predikat PLATIO:
PLATIO(Toma),
PLATIO(Jova),
PLATIO(Mirko)

Sopstvenom aksiomom može se uvesti predikat DUŽNIK:

$$\text{DUŽNIK}(X) \Leftrightarrow \neg\text{PLATIO}(X)$$

(u konkretnom slučaju može se uzeti jedna od implikacija).

Tako uvedeni predikati mogu ali i ne moraju biti deklarisani kao CWA-predikati. U stvari, ako je predikat iz baze CWA to se posredno odražava na ostale predikate koji su sa njim u vezi. Zato rezolucijska procedura obrađuje te predikate dokle god može, odnosno dok ne dođe do CWA-predikata koje dalje ne može da rezolvira, pa se za njihovu obradu primenjuje CWA-pravilo.

Primer 4.5. Posmatraju se podaci iz ***primera 4.4.*** i sopstvena aksioma:

$$\neg PLATIO(X) \Rightarrow DUŽNIK(X)$$

koja u internoj formi BASELOG-a ima sledeći izgled:

PLATIO(X1)DUZNIK(X1)&

i neka je definisan CWA-predikat:
CWA[1]:='PLATIO'

Ako se postavi pitanje:
Da li je Žarko dužnik?

odnosno u internoj formi BASELOG-sistema:
~DUZNIK(Zarko)&

Tok dokaza ima sledeći izgled:

nivo =1, centralni sastavak: ~DUZNIK(Zarko)&
rezolventa: /~DUZNIK(Zarko)PLATIO(Zarko)&

nivo = 2, centralni sastavak: /~DUZNIK(Zarko)PLATIO(Zarko)&

ulaz u CWA-kontroler: /~DUZNIK(Zarko)PLATIO(Zarko)&
predikat=PLATIO
ulaz u proceduru **ibris**: /~DUZNIK(Zarko)PLATIO(Zarko)&
ulaz u proceduru **isaz**: /~DUZNIK(Zarko)
ulaz u proceduru **iskr**: /~DUZNIK(Zarko)
nova rezolventa: &
DOKAZ JE NAĐEN

Odgovor BASELOG-sistema je da je Žarko dužnik.

Na ovom primeru se vidi da je za CWA-predikat 'PLATIO', koji se nalazi u bazi podataka, aktiviran CWA-kontroler nakon što je rezolucijska procedura obradila predikate iz programa i nije uspela da generiše novu rezolventu.

4.4. CWA-kontroler kao modul za proširenje ADT sistema

Pravilo CWA je iskorišćeno za razvoj modula kojim se proširuje postojeći ADT sistem, koji je zasnovan na OL rezoluciji sa markiranim literalima, što je opisano u odeljku 3.1.2.

CWA-kontroler za ovu vrstu rezolucije radi na sledeći način:

- *U slučaju da poslednji literal centralnog sastavka, koji je oblika $R(t_1,...,t_n)$ nije mogao biti pobijen pomoću sopstvenih aksioma, i nije mogao biti unificiran ni sa jednim elementom iz baze, na osnovu CWA-pravila briše se taj literal.*

  Na tako identifikovan sastavak primenjuju se operacije:
  *- **skraćivanja** (brisanje poslednjeg nemarkiranog literala), i*
  *- **sažimanja** (brisanje poslednjeg nemarkiranog literala identičnog nekom prethodnom nemarkiranom literalu i ispitivanje tautologičnosti).*

  Ove operacije se inače primenjuju u procesu generisanja OL rezolvente.
  Tako određeni sastavak predstavlja novu rezolventu koja se uzima u svojstvu novog centralnog sastavka za generisanje rezolvente na narednom nivou.

Napomena: Pojmovi: *markirani, nemarkirani literal, operacije* ***skraćivanja, sažimanja*** detaljnije su objašnjeni u odeljku 3.1.2.

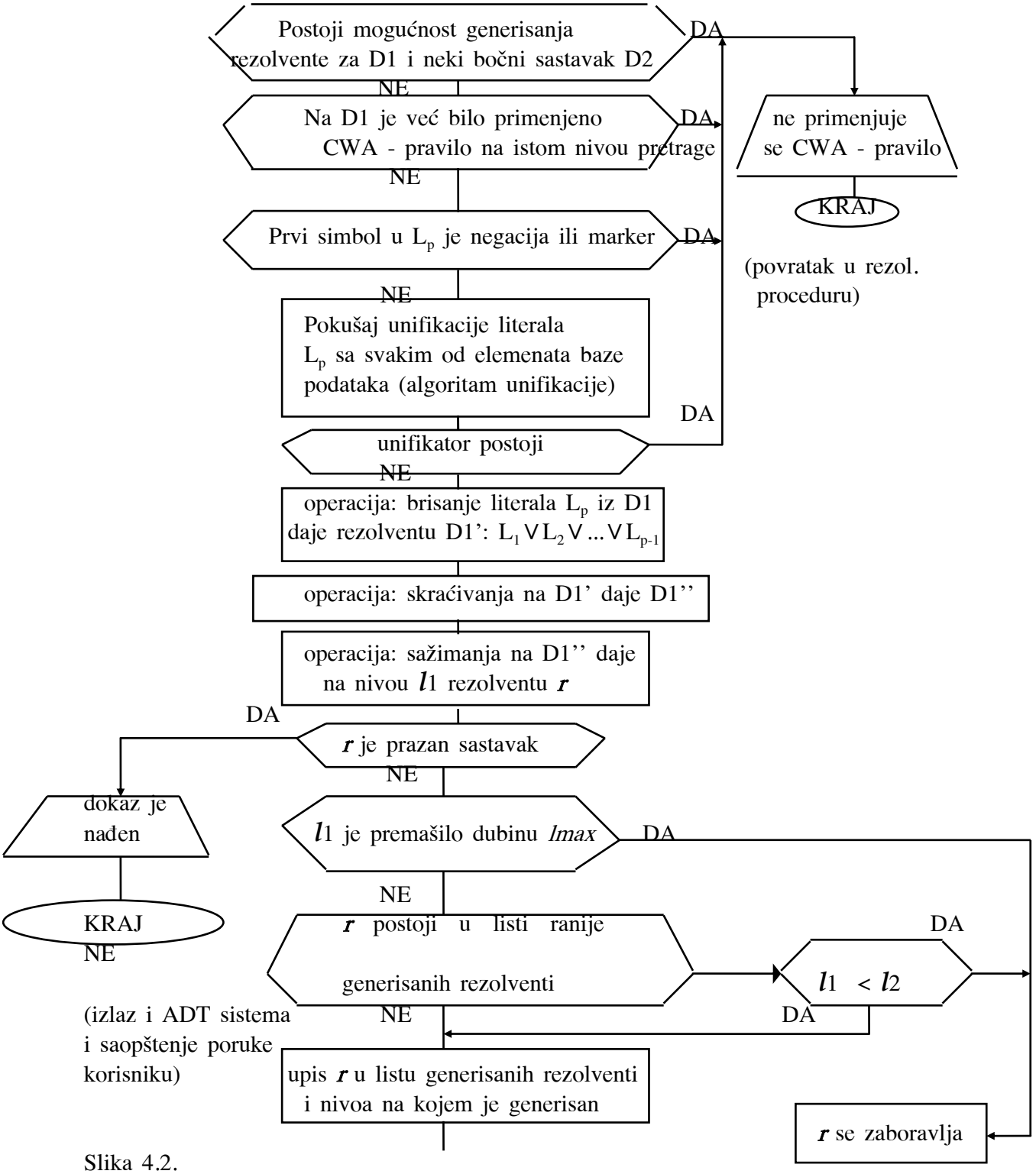
To znači da se *tekući sastavak D obrađuje rezolucijom dokle god je to moguće*, a tek kad se nova rezolventa ne može generisati, onda se primenjuje CWA-pravilo.

Sam ***CWA-kontroler*** se kao samostalan modul aktivira samo za one predikate, koji su definisani u listi CWA-predikata. Za sve ostale predikate sistem radi kao ADT sistem bez CWA-kontrolera.

Modul sadrži sledeće procedure:

- procedura kojom se realizuje algoritam unifikacije,
- procedura za brisanje poslednjeg literala iz centralnog sastavka,
- procedura za realizaciju operacije skraćivanja,
- procedura za realizaciju operacije sažimanja, i
- procedura kojom se vrši provera postojanja identičnih rezolventi.

Šematski prikaz CWA-kontrolera za OL rezoluciju sa markiranim literalima dat je na slici 4.2.



Slika 4.2.

4.5. BASELOG-sistem i njegove mogućnosti

BASELOG-sistem je sistem ADT koji je proširen modulom koji se zove CWA-kontroler. Osnovni koncepti BASELOG-sistema su:

- Aksiome tipa BAKS, putem kojih su u sistemu predstavljeni elementi baze podataka (torke)
- Sopstvene aksiome su aksiome tipa AKS
- Lista CWA-predikata
- Upiti

Aksiome tipa BAKS služe za predstavljanje elemenata baze podataka. To znači da svaki element baze podataka dobija status aksiome. Za datu bazu, elementi BAKS-a se formiraju na sledeći način. Naziv šeme relacije uzima se kao ime predikata. Argumenti predikata su obeležja šeme relacije. Ograničenja koja su prisutna u šemi relacije mogu se opisati pomoću posebnih aksioma za svako ograničenje.

Primer 4.6. Data je sledeća šema relacije:

STUDENT({br_ind,ime, prezime, god_stud},{γ}),
gde obeležje *br_ind* ima značenje - broj indeksa studenta a obeležje *god_stud* ima značenje - godina studija studenta. Sa γ je označeno ograničenje u ovoj šemi relacije koje glasi:

- svaki student ima samo jedan broj indeksa i ne postoje dva studenta sa istim brojem indeksa.

Jedna pojava nad ovom šemom relacije je sledeća relacija:

BR_IND	IME	PREZIME	GOD_STUD
160	Marko	Marić	2
012	Ivan	Jovanović	3
114	Mira	Ilić	1

Ova relacija je predstavljena u BASELOG-sistemu putem sledećih aksioma tipa BAKS:

student(160,Marko,Maric,2)
student(012,Ivan,Jovanovic,3)
student(114,Mira,Ilic,1)

Ograničenje γ se može opisati sledećom aksiomom:

$$(Student(x_1,y_1,z_1,u_1) \wedge Student(x_2,y_2,z_2,u_2) \wedge x_1 = x_2) \Rightarrow y_1 = y_2 \wedge z_1 = z_2 \wedge u_1 = u_2$$

S obzirom da u praksi formiranja baza podataka sami programi za ažuriranje uzimaju u obzir ova ograničenja, takve aksiome nisu neophodne za rad BASELOG-a. Ako je reč o ograničenjima koja nisu uključena u mehanizme održavanja integriteta podataka u bazi, onda BASELOG omogućuje unošenje takvih ograničenja putem njihovog opisa u vidu posebnih aksioma. Za to je potrebno samo izvršiti adekvatnu formalizaciju takvih ograničenja. Sve takve aksiome imaju status *sopstvenih aksioma* u BASELOG-u.

Sopstvene aksiome tipa AKS služe da se na sličan način, kao u DATALOG-u, predstave aksiome u programu. (Putem EDB predikata se u DATALOG-u vrši predstavljanje skupa osnovnih činjenica fizički pohranjenih u relacionoj bazi podataka, a sam DATALOG program sadrži IDB predikate. Ovi koncepti su detaljnije objašnjeni u odeljku 3.3.1.)

Prema tome, sopstvene aksiome u BASELOG-u uvode nove predikate spregnute sa predikatima iz baze ili opisuju željena ograničenja i druge uslove, koji se zahtevaju od konkretne baze.

Lista CWA-predikata sadrži niz predikata (naziva predikata) iz programa koje želimo da deklarišemo za CWA-predikate, koje će sistem obraditi u režimu zatvorenog sveta (samo za te predikate je svet zatvoren).

Upiti se postavljaju od strane korisnika i u BASELOG-u imaju sledeću internu formu: $\neg R(w_1,...,w_m)$, gde je R - naziv predikata, a $w_1,...,w_m$ su argumenti (konstante ili promenljive). Upite je moguće postaviti u prirodnom obliku - onako kako ih korisnik zadaje. Interfejs, razvijen za tu svrhu prevodi korisnikov upit u interni oblik BASELOG-sistema.

Upitni predikat R može biti deklarisan za CWA-predikat ili ne.

BASELOG-sistem je, kao što je napred rečeno, sistem ADT proširen CWA-kontrolerom. Kako se ADT sistem zasniva na metodi rezolucije potrebno je **negirati** formulu koja se dokazuje. U BASELOG-sistemu to znači da se polazi od **negacije upita**.

Negacija upita, sastavci iz aksioma tipa BAKS i sastavci iz sopstvenih aksioma čine polazni skup sastavaka za OL rezoluciju. Lista CWA-predikata i aksiome tipa BAKS su komponente pomoću kojih se realizuje rad CWA-kontrolera u ADT sistemu.

Rad BASELOG-sistema završava se odgovorom na upit, odnosno porukom korisniku.

Veza internog rezultata rada sistema sa semantičkim značenjem se određuje na sledeći način:

- Ako je **upitni predikat deklarisan kao CWA-predikat,** onda interni odgovori BASELOG-sistema imaju sledeće značenje:

1. Odgovor sistema je **uspeh = 1**, to znači potvrdan odgovor na upit odn. "**DA**".
2. Odgovor sistema je **uspeh = 0** i nema više novih rezolventi, onda to znači negativan odgovor na upit, odnosno "**NE**".
3. Odgovor sistema je **uspeh = 0** i moguće je generisati još rezolventi, ali se proces prekida usled prostorno - vremenskih ograničenja, onda to znači da je odgovor sistema "**neizvesno**".

- Ako **upitni predikat nije deklarisan kao CWA-**predikat, onda se odgovori BASELOG-sistema tumače isto kao u ADT sistemu na sledeći način:

1. Odgovor sistema je **uspeh = 1**, to znači potvrdan odgovor na upit odn. "**DA**".
2. Odgovor sistema je **uspeh = 0** i nema više novih rezolventi to znači da "**nije izvodivo"** (nije dokazivo), ali ne znači NE (NIJE).
3. Odgovor sistema je **uspeh = 0** i moguće je generisati još rezolventi, ali se proces prekida usled prostorno - vremenskih ograničenja, onda je odgovor sistema "**neizvesno"**.

Ovo su odgovori koji odgovaraju radu ADT sistema.

Primeri rada BASELOG-sistema

Razne moguće varijante odgovora BASELOG-sistema ilustrovane su sledećim primerima

Primer 4.7. Neka je data relacija nad šemom PLATIO(ime) čija je semantika spisak imena građana koji su platili račun.

PLATIO(Toma)
PLATIO(Jova)
PLATIO(Mirko)

U ovom primeru se pretpostavlja da ne postoje sopstvene aksiome.
Neka je postavljen sledeći upit nad ovom bazom podataka:

Da li je Žarko platio račun?

U internoj formi BASELOG-sistema ovaj upit ima sledeći oblik (polazi se od negacije upita, simbol za negaciju je "~"):

~PLATIO(Zarko)&

Odgovori BASELOG-sistema su prikazani u tabeli 4.1. a detaljnije objašnjenje u tekstu koji sledi.

Pitanje:	Da li je Žarko platio račun?	
	predikat PLATIO **je CWA-predikat** (**Slučaj A.**)	predikat PLATIO **nije CWA-predikat** (**Slučaj B.**)
činjenica PLATIO(žarko) ne postoji u bazi	uspeh = 0 (NE)	uspeh = 0 (nije izvodivo)
činjenica PLATIO(žarko) postoji u bazi	uspeh = 1 (DA)	uspeh = 1 (DA)

Tabela 4.1.

Slučaj A. Upitni predikat PLATIO je deklarisan kao CWA-predikat (režim zatvorenog sveta)

1. S obzirom da činjenica PLATIO(Zarko) ne postoji u bazi podataka (među aksiomama tipa BAKS) onda sistem daje odgovor **uspeh = 0** što se tumači kao semantički odgovor "**ne**" na postavljeni upit, odnosno odgovor sistema je: Žarko nije platio račun.
U stvari, ovde sistem BASELOG ne može da generiše rezolventu sa elementima baze, a zbog prisustva negacije predikata PLATIO neće primeniti brisanje u CWA-kontroleru.
2. Ako činjenicu PLATIO(Zarko) unesemo u bazu podataka sistem izvodi **uspeh = 1** što se tumači kao odgovor "**da**", odnosno Žarko je platio račun.
Taj odgovor sistem će izvesti rezolucijom bez CWA-kontrolera.

Slučaj B. Upitni predikat PLATIO nije deklarisan kao CWA-predikat (režim rada sistema ADT)

1. S obzirom da činjenica PLATIO(Zarko) ne postoji u bazi podataka onda sistem izvodi **uspeh = 0** i saopštava odgovor "**nije izvodivo**", što ne znači da Žarko nije platio račun.
2. Ako činjenicu PLATIO(Zarko) unesemo u bazu podataka sistem izvodi **uspeh = 1** što se tumači kao odgovor "**da**".

Primer 4.7.1. Neka je u odnosu na bazu iz ***Primera 4.7.*** postavljeno pitanje:

"Da li Žarko nije platio račun?"

U internoj formi BASELOG-sistema ovaj upit ima sledeći oblik (negacija upita):

PLATIO(Zarko)&

Odgovori BASELOG-sistema su prikazani u tabeli 4.2. a detaljnije objašnjenje u tekstu koji sledi.

Pitanje:	Da li Žarko nije platio račun?	
	predikat PLATIO **je CWA-predikat** (**Slučaj A.**)	predikat PLATIO **nije CWA-predikat** (**Slučaj B.**)
činjenica PLATIOZarko) ne postoji u bazi	uspeh = 1 (DA)	uspeh = 0 (nije izvodivo)
činjenica PLATIO(Zarko) postoji u bazi	uspeh = 0 (NE)	uspeh = 0 (nije izvodivo)

Tabela 4.2.

Slučaj A. Upitni predikat PLATIO je deklarisan kao CWA-predikat (režim zatvorenog sveta)

1. S obzirom da činjenica PLATIO(Zarko) ne postoji u bazi podataka (među aksiomama tipa BAKS) onda sistem izvodi **uspeh = 1** što se tumači kao odgovor "**da**" na upit odnosno odgovor sistema korisniku je: Da, Žarko nije platio račun.
Ovaj odgovor sistem izvodi aktiviranjem CWA-kontrolera, primenom CWA-pravila, s obzirom da rezolucija sa elementima baze nije moguća.
2. Ako činjenicu PLATIO(Zarko) unesemo u bazu podataka sistem izvodi odgovor **uspeh = 0**, što znači da je odgovor na upit "**ne**". Odgovor sistema korisniku je: Nije tačno da Žarko nije platio račun, znači - Žarko je platio račun.
U ovom slučaju rezolucija nije moguća, a primena CWA-pravila ne dovodi do brisanja literala PLATIO(Zarko), jer je moguća unifikacija upitnog predikata sa aksiomom iz baze.

Slučaj B. Upitni predikat PLATIO nije deklarisan kao CWA-predikat (režim rada sistema ADT)

1. S obzirom da činjenica PLATIO(Zarko) ne postoji u bazi podataka sistem izvodi odgovor **uspeh= 0**, što znači "**nije izvodivo**", pa je odgovor sistema: Ne može se tvrditi (dokazati) da Žarko nije platio račun. Sama rezolucijska procedura ne može to da dokaže.
2. Ako činjenicu PLATIO(Zarko) unesemo u bazu podataka sistem izvodi odgovor **uspeh = 0,** što se tumači na isti način kao pod 1.

Primer 4.8. Neka je data pojava nad šemom relacije kao i u ***Primeru 4.7.***
PLATIO(Toma)
PLATIO(Jova)
PLATIO(Mirko)
i neka je definisana i sledeća sopstvena aksioma:

$$\neg \text{PLATIO}(X) \Rightarrow \text{DUŽNIK}(X)$$

U internoj formi BASELOG-a ona ima sledeći oblik:

$$\text{PLATIO}(X) \vee \text{DUŽNIK}(X)$$

Sada se može postaviti sledeći upit nad ovom bazom podataka:

Da li je Žarko dužnik?

U internoj formi BASELOG-sistema ovaj upit ima sledeći oblik:

~DUZNIK(Zarko)&

Odgovori BASELOG-sistema su prikazani u tabeli 4.3. a detaljnije objašnjenje u tekstu koji sledi.

Pitanje:	Da li je Žarko dužnik?	
	predikat PLATIO **je CWA-predikat** (**Slučaj A.**)	predikat PLATIO **nije CWA-predikat** (**Slučaj B.**)
činjenica PLATIO(Zarko) ne postoji u bazi	uspeh = 1 (DA)	uspeh = 0 (nije izvodivo)
činjenica PLATIO(Zarko) postoji u bazi	uspeh = 0 (NE)	uspeh = 0 (nije izvodivo)

Tabela 4.3.

Slučaj A. Predikat PLATIO je deklarisan kao CWA-predikat (režim zatvorenog sveta za predikat PLATIO)

1. S obzirom da činjenica PLATIO(Zarko) ne postoji u bazi podataka sistem daje odgovor **uspeh = 1** što se tumači kao semantički odgovor "**da**" na postavljeni upit, odnosno odgovor sistema je da je Žarko dužnik.
Do takvog odgovora sistem dolazi na sledeći način:

Rezolucijom upita sa sopstvenom aksiomom izvodi se rezolventa PLATIO(Zarko). Nema više novih rezolventi pa CWA-kontroler primenom CWA-pravila briše predikat PLATIO i izvodi prazan sastavak.
2. Ako činjenicu PLATIO(Zarko) unesemo u bazu podataka sistem izvodi **uspeh = 0** što se tumači kao odgovor "**ne**", odnosno odgovor sistema je: Ne, Žarko nije dužnik.
Ovaj odgovor sistem generiše na sledeći način:
Nakon što rezolucijom upita sa sopstvenom aksiomom izvede rezolventu PLATIO(Zarko) CWA-kontroler ne može da izvrši brisanje i izvede prazan sastavak, jer je moguća unifikacija sa elementom u bazi.

Slučaj B. Lista CWA-predikata je prazna (režim rada sistema ADT)
Za interni oblik upita ~DUZNIK(Zarko) nakon izvođenja rezolvente PLATIO(Zarko) biće:

1. S obzirom da činjenica PLATIO(Zarko) ne postoji u bazi podataka onda sistem izvodi **uspeh = 0** i saopštava odgovor "**nije izvodivo**", pa je odgovor sistema: Ne može se tvrditi (dokazati) da je Žarko dužnik.
2. Ako činjenicu PLATIO(Zarko) unesemo u bazu podataka sistem izvodi odgovor **uspeh = 0,** što se tumači na isti način kao pod 1.
U oba slučaja nema novih rezolventi, a CWA-kontroler se ne uključuje, jer nema deklarisanih CWA-predikata.

4.5.1. Korisnički aspekt - zadavanje upita i odgovori sistema

Razvojem posebnog interfejsa može se obezbediti visok komfor za rad korisnika bez potrebe za poznavanjem unutrašnjih mehanizama BASELOG-sistema. Taj interfejs treba da obezbedi povezivanje internih odgovora u sistemu sa semantičkim odgovorima koji se saopštavaju korisniku. To je predstavljeno u tabeli 4.4.

Interni rezultat rada BASELOG-sistema	**Odgovor na upit korisniku**	
	Upitni predikat je CWA	Upitni predikat nije CWA
1. Uspeh = 1	potvrdan ("DA")	potvrdan ("DA")
2. Uspeh = 0 i nema više rezolventi	odričan ("NE" - "NIJE")	"NIJE IZVODIVO" (ne mora da znači NE)
3. Uspeh = 0 i proces generisanja je prekinut	"NEIZVESNO"	"NEIZVESNO"

Tabela 4.4.

Na osnovu tabele 4.4. jasno se uočava bitna razlika u reagovanju sistema sa i bez CWA-kontrolera za slučaj pod 2.

Ako se radi o klasičnoj bazi podataka, onda praksa zahteva decidan odgovor, što se obezbeđuje radom sa CWA-kontrolerom (na primer, *platio - nije platio*).

Ako se radi o bazi znanja u obrazovnom računarskom softveru, onda odgovor sa CWA ne bi bio prihvatljiv, već je prihvatljiv odgovor u režimu bez CWA (U ***primeru 4.1.*** za *paralelnost pravih* iz odeljka 4.1., uz CWA, režim daje neprihvatljiv odgovor: "A nije paralelno sa C", dok je prihvatljiv odgovor: "*nije izvodivo*", u režimu bez CWA, za predikat "paralelno").

Treba naglasiti, da je bitna razlika ne samo u slučaju pod 2. nego i u postojanju mogućnosti da se uz CWA dobije uspeh=1, dok se bez CWA uspeh=1 ne može dobiti.

Primer 4.9. Data je relacija PLATIO(ime) čija je semantika spisak imena građana koji su platili račun:

PLATIO(Toma)
PLATIO(Jova)
PLATIO(Mirko)
i neka je definisana i sledeća sopstvena aksioma:

$$\neg \text{PLATIO(X)} \Rightarrow \text{DUŽNIK(X)}$$

U internoj formi BASELOG-a ona ima sledeći oblik:

$$\text{PLATIO(X)} \vee \text{DUŽNIK(X)}$$

Ako se postavi upit nad ovom bazom podataka:

Da li je Žarko dužnik?
odnosno u internoj formi BASELOG-sistema:

~DUZNIK(Zarko)&

i ako je lista CWA-predikata prazna sistem daje sledeći odgovor:
S obzirom da činjenica PLATIO(Zarko) ne postoji u bazi podataka onda sistem izvodi **uspeh = 0** i saopštava odgovor "**nije izvodivo**", pa je odgovor sistema: Ne može se tvrditi (dokazati) da je Žarko dužnik.

Ako se u listu CWA-predikata unese predikat PLATIO odgovor BASELOG-sistema je: **uspeh = 1** što se tumači kao semantički odgovor "**da**" na postavljeni upit, odnosno odgovor sistema je da je Žarko dužnik.
Do takvog odgovora sistem dolazi na sledeći način:
Rezolucijom upita sa sopstvenom aksiomom izvodi se rezolventa PLATIO(Zarko). Nema više novih rezolventi pa CWA-kontroler primenom CWA-pravila briše predikat PLATIO i izvodi prazan sastavak.

Zato je važno dobro odrediti semantički aspekt upita i u vezi sa tim deklarisati koji predikati treba da budu CWA a koji ne.

Za svaku konkretnu bazu, mogu se unapred odrediti i fiksirati predikati koji treba da budu CWA i oni koji treba da ostanu slobodni. To ne mora da radi korisnik, mada se korisniku može, kao što je to u BASELOG-u učinjeno, ostaviti mogućnost da i ***sam*** deklariše CWA-predikate.

Projektant nastavne baze podataka u saradnji sa nastavnikom definiše listu CWA-predikata. Preporuke za definisanje liste CWA-predikata su:

- ako je potrebno bazu podataka posmatrati kao zatvoren svet, onda lista CWA-predikata treba da sadrži sve predikate koji se pojavljuju u bazi podataka. To je slučaj kod baze podataka koja sadrži definicije iz neke oblasti ili se koristi za testiranje znanja učenika. Odgovor na upit učenika u tom slučaju treba da bude decidan: "Da (Ne)" ili "Odgovor je tačan (netačan)."
  Na primer, ako na postavljeno pitanje "Koliki je zbir uglova u trouglu?" učenik odgovori "360^{o}", sistem na osnovu podataka u bazi treba učeniku da saopšti da je učenikov odgovor netačan.

- U slučaju da učenik koristi bazu podataka radi dobijanja informacija ili sticanja novih znanja, onda baza podataka u kojoj nisu sadržana sva relevantna znanja treba da ima semantiku otvorenog sveta, i u tom slučaju predikati koji se pojavljuju u bazi moraju imati status predikata za koje sistem radi u režimu otvorenog sveta. To je zato da bi se omogućilo korisniku da slobodno postavlja upite, a da odgovori sistema na takve upite ne dovedu korisnika u zabludu (bolje je da odgovori budu neodređeni, nego pogrešni).

Na primer, ako je u bazi definisana relacija ***posle***, ali se ne nalaze podaci o boju na Čegru i boju na Marici, onda učenik može da postavi pitanje: "Da li je boj na Čegru bio posle boja na Marici?". Odgovor sistema će biti "**neizvesno**", dok ako bi "**posle**" bio deklarisan kao CWA-predikat onda bi odgovor sistema bio "**NE**" što bi bila dezinformacija.

4.6. Komparacija BASELOG-sistema sa ADT, DATALOG-om i PROLOG-om

4.6.1. Komparacija BASELOG-a sa ADT

BASELOG-sistem je proširenje u odnosu na ADT putem liste CWA-predikata, CWA-pravila i CWA-kontrolera. Ako je lista CWA-predikata prazna BASELOG-sistem radi u režimu ADT sistema odnosno kao univerzalni rezolucijski sistem zasnovan na OL rezoluciji sa markiranim literalima. Proširenje u odnosu na ADT sistem dolazi do izražaja samo kada lista CWA-predikata nije prazna. Ako lista CWA-predikata nije prazna, onda BASELOG-sistem eliminiše potrebu pisanja dodatnih aksioma za ADT sistem (koje je u praksi često nemoguće efektivno zapisati zbog njihove dužine). Zato je BASELOG proširenje ADT sistema.

Sledeći primer ilustruje situaciju u kojoj BASELOG-sistem radi kao proširenje ADT sistema, putem liste CWA-predikata, kojom se zamenjuje jedna dugačka aksioma.

Primer 4.10. U radu [LHRB95] je opisan problem zadovoljenja generalizovanih zavisnosti podataka (E i T zavisnosti) metodom automatskog rezonovanja - upotrebom ADT sistema. Na ovom mestu uzeće se kao primer samo dokaz zadovoljenja T - zavisnosti. T-zavisnost je formulisana u [ML96].

Za dokaz da T-zavisnost $<\tau, \ell>$, gde je $\tau = \{(x_1,y_1,z_1,w_1), (x_1,y_2,z_2,w_2)\}$ i $\ell = (x_1,y_1,z_2,w_2)$, u relaciji r nije zadovoljena pomoću putem ADT sistema definisan je sledeći polazni skup sastavaka S:

1. $\neg P(x_1,y_1,z_1,w_1) \vee \neg P(x_1,y_2,z_2,w_2) \vee P(x_1,y_1,z_2,w_2)$ (teorema o T - zavisnosti)

Torke relacije r, kao elementi baze podataka, predstavljene su putem predikata P:

2. $P(a_1,b_1,c_1,d_1)$
3. $P(a_2,b_1,c_2,d_2)$
4. $P(a_1,b_2,c_1,d_1)$
5. $P(a_1,b_2,c_2,d_2)$

Dugačka aksioma kojom se zadaje kompletan opis prostora pretrage je oblika:

6. $(x,y,z,w) = (a_1,b_1,c_1,d_1) \vee (x,y,z,w) = (a_2,b_1,c_2,d_2) \vee (x,y,z,w) = (a_1,b_2,c_1,d_1) \vee$
   $\vee (x,y,z,w) = (a_1,b_2,c_2,d_2) \vee \neg P(x,y,z,w)$

U implikativnoj formi aksiome 1. i 6. glase:
1. $(P(x_1,y_1,z_1,w_1) \wedge P(x_1,y_2,z_2,w_2)) \Rightarrow P(x_1,y_1,z_2,w_2)$ (uslov T - zavisnosti)

2. $((x,y,z,w) \neq (a_1,b_1,c_1,d_1) \wedge (x,y,z,w) \neq (a_2,b_1,c_2,d_2) \wedge (x,y,z,w) \neq (a_1,b_2,c_1,d_1) \wedge$
   $\wedge (x,y,z,w) \neq (a_1,b_2,c_2,d_2)) \Rightarrow \neg P(x,y,z,w)$

Na osnovu ovako definisanog skupa uslova sistem ADT je generisao prazan sastavak, što znači da relacija ***r***, koja je predstavljena putem datih torki, ne zadovoljava T - zavisnost, što je i trebalo dokazati. □

Kao što se na osnovu ovog primera vidi, aksioma za zadavanje kompletnog prostora pretrage već i za mali broj torki u bazi (u ovom slučaju samo 4) ima veliku dužinu, a to je upravo i bio glavni razlog zbog kojeg se odustalo od njene praktične primene (kao što je to i objašnjeno u odeljku 4.1.).

Sada će se pokazati kako se ovaj isti primer realizuje upotrebom BASELOG-sistema, ali putem liste CWA predikata, kojom se eliminiše pisanje dugačke aksiome.

Lista CWA predikata sadrži samo jedan predikat 'P', i za njega BASELOG-sistem radi u režimu zatvorenog sveta.

Elementi baze podataka su predstavljeni putem predikata P i to su aksiome tipa BAKS (u internoj formi BASELOG-a). Aksiome 1.-5. su iste, dok dugačka aksioma 6. nije potrebna.

BASELOG-sistem generiše prazan sastavak uz sledeće parametre koji definišu okvir pretraživanja:
- maksimalna (granična) dubina pretraživanja: 10 nivoa,
- maksimalna dužina rezolvente: 100 karaktera.

U slučaju da je dokaz nađen sistem daje korisniku sledeće informacije:
uspeh = 1
broj generisanih rezolventi - ovaj broj označava ukupan broj rezolventi koje je rezolucijska procedura generisala u toku izvođenja dokaza
maksimalno dostignuti nivo - maksimalna dubina dokaza do koje je došla rezolucijska procedura u toku nalaženja dokaza. Ukupan broj nivoa je parametar koga korisnik može da menja i na taj način utiče na tok dokaza
nivo na kojem je generisan prazan sastavak

Na ovom primeru sistem daje sledeću poruku:
Uspeh =1
dokaz nađen
broj generisanih rezolventi = 17
maksimalno dostignuti nivo = 10
nivo na kojem je generisan prazan sastavak = 10

DOKAZ:

nivo = 1; centralni sastavak: P(X1,Y1,Z2,V2)~P(X1,Y2,Z2,V2) ~P(X1,Y1,Z1,V1)&
2. bočni, 1. literal:
P(a1,b1,c1,d1)&

nivo = 2; rezolventa:
P(a1,b1,Z2,V2) ~P(a1,Y2,Z2,V2)&
2. bočni, 1. literal:
P(a1,b1,c1,d1)&

nivo = 3; rezolventa:
P(a1,b1,c1,d1)&
1. bočni, 2. literal:
P(X1,Y1,Z2,V2)~P(X1,Y2,Z2,V2)~P(X1,Y1,Z1,V1)&

nivo = 4; rezolventa:
/P(a1,b1,c1,d1)P(a1,Y1,c1,d1)~P(a1,Y1,Z1,V1)&

nivo = 5; rezolventa prethodnika: /P(a1,b1,c1,d1)P(a1,b1,c1,d1)&
1. bočni, 2. literal:
P(X1,Y1,Z2,V2)~P(X1,Y2,Z2,V2)~P(X1,Y1,Z1,V1)&

nivo = 6; rezolventa:
/P(a1,b1,c1,d1)/P(a1,b1,c1,d1)P(a1,Y1,c1,d1)~P(a1,Y1,Z1,V1)&

nivo = 7; rezolventa prethodnika: /P(a1,b1,c1,d1)/P(a1,b1,c1,d1)P(a1,b1,c1,d1)&
1. bočni, 3. literal:
P(X1,Y1,Z2,V2)~P(X1,Y2,Z2,V2)~P(X1,Y1,Z1,V1)&

nivo = 8; rezolventa:
/P(a1,b1,c1,d1)/ P(a1,b1,c1,d1)/ P(a1,b1,c1,d1)P(a1,b1,Z2,V2)~P(a1,Y2,Z2,V2)&
5. bočni, 1. literal:
P(a1,b2,c2,d2)&

nivo = 9; rezolventa:
/P(a1,b1,c1,d1)/ P(a1,b1,c1,d1)/ P(a1,b1,c1,d1)P(a1,b1,c2,d2)&
brisanje po CWA-pravilu

nivo = 10; rezolventa
&
DOKAZ JE ZAVRŠEN

Na osnovu ovog primera vidi se da je putem liste CWA-predikata u BASELOG-sistemu eliminisana potreba za pisanjem dugačke aksiome, koja je bila neophodna za rad ADT sistema, pa na osnovu ovoga zaključuje da je BASELOG-sistem proširenje u odnosu na ADT sistem u smislu istog efekta, ali uz manje zahteve.

4.6.2. Komparacija BASELOG-a sa DATALOG-om

DATALOG radi u režimu zatvorenog sveta kao što je opisano u odeljku 3.3.1. BASELOG-sistem, je kao što je to napred objašnjeno, uopštenje u odnosu na rad DATALOG-a, jer je moguć rad i u otvorenom, u zatvorenom i u poluotvorenom - poluzatvorenom režimu rada. Ovaj poslednji režim rada znači da su samo neki predikati deklarisani kao CWA-predikati ali ne svi predikati iz programa.

Stav 4.1. Ako su svi predikati deklarisani kao CWA-predikati, koji su navedeni u listi CWA, onda BASELOG-sistem radi kao DATALOG sistem.

Dokaz.

Neka je na nivou l BASELOG-sistem generisao sledeći centralni sastavak:

$$D1: \quad L_1 \vee \ldots \vee L_p,$$

gde su L_i - literali, L_p: $R(w_1,\ldots,w_m)$, gde je R predikat (naziv predikata). $R \in CWA$, gde je CWA skup svih predikata koji se pojavljuju u programu.

Tada su za D1 moguće sledeće tri opcije:

1) Iz D1 se rezolucijom prethodnika izvodi rezolventa prethodnika - novi centralni sastavak na nivou l+1, oblika:

a) $D: \quad L_{1\theta} \vee \ldots \vee L_{s\theta}, \quad 1 \leq s \leq p-1,$

ili

b) D je prazan sastavak

(θ je zamena rezolucije prethodnika, videti u odeljku 3.1.2.)

2) Iz D1 i D2 OL rezolucijom se izvodi rezolventa - novi centralni sastavak na nivou l+1.
Neka je bočni literal D2 oblika:

$$Q_1 \vee \ldots \vee Q_k \ldots \vee Q_r, \quad 1 \leq k \leq r$$

takav da za zamenu θ važi poklapanje $Q'_{k\theta}$ sa $L_{p\theta}$, pri čemu je Q_k komplementaran u odnosu na negaciju literalu L_p (Q'_k je Q_k bez simbola negacije). Q_k je oblika $\neg R(u_1,...,u_m)$, $u_1,...,u_m$ su termi, i

$$Q'_{k\theta} \equiv L_{p\theta} \equiv R(u_{1\theta},...,u_{m\theta}) \equiv R(w_{1\theta},...,w_{m\theta}).$$

Tada je rezolventa:

$$D: L_{1\theta} \vee ... \vee L_{k\theta} \vee / L_{p-1\theta} \vee Q_{1\theta} \vee ... \vee Q_{k-1\theta} \vee Q_{k+1\theta} \vee ... Q_{r\theta}$$

3) Ne postoji mogućnost izvođenja nove rezolvente na l+1 nivou.
Sledi, ne postoji bočni sastavak D2 sa k-tim literalom Q_k, koji je komplementaran literalu L_p ili ne postoji unifikator θ za Q'_k i L_p.

U opcijama pod 1) i 2) oba sistema rade bez korišćenja CWA i na osnovu potpunosti sistema logičkog izvođenja dovode do istih rezultata. Preostaje za dokaz samo slučaj 3) kad izvođenje rezolvente nije moguće.

U opciji 3) u programu ne postoji literal oblika $\neg R(u_1,...,u_m)$ koji se za neko θ ujednačava sa $R(w_1,...,w_m)$. Literal $R(w_1,...,w_m)$ se ne može pobiti rezolucijskom procedurom, pa sledi da $\neg R(w_1,...,w_m)$ nije dokazivo (izvodivo) iz BASELOG programa rezolucijom ($\neg R(w_1,...,w_m)$ nije teorema, odnosno nije logička posledica aksioma), bez principa CWA.

Kad $\neg R(w_1,...,w_m)$ nije logička posledica aksioma, onda za $R(w_1,...,w_m)$ postoje dve mogućnosti:

3a) $R(w_1,...,w_m)$ jeste logička posledica aksioma,
3b) $R(w_1,...,w_m)$ nije logička posledica aksioma.

U takvoj situaciji za literal $R(w_1,...,w_m)$ postoje dve moguće opcije:

3a) Postoji unifikator za $R(w_1,...,w_m)$ i neki j-ti element baze ($1 \leq j \leq n$, n je ukupan broj torki (elemenata) baze).
3b) Ne postoji unifikator za $R(w_1,...,w_m)$ za j-ti element baze, $\forall j$, ($j \in \{1,...,n\}$).

Slučaj 3a). Neka je θ unifikator za $R(w_1,...,w_m)$ i j-ti element baze $R(u_1,...,u_m)$, odnosno

$$R(w_{1\theta},...,w_{m\theta}) \equiv R(u_{1\theta},...,u_{m\theta})$$

Rezolucija nije moguća, jer literali nisu komplementarni. Literal $R(w_1,...,w_m)$ je izvodiv iz aksioma unifikacijom, pa je $R(w_1,...,w_m)$ logička posledica aksioma i kao takva se ne može pobiti, ($\neg R(w_1,...,w_m)$ se može pobiti).

Dakle, u ovom slučaju $R(w_1,...,w_m)$ jeste logička posledica aksioma i $\neg R(w_1,...,w_m)$ nije logička posledica aksioma.

U BASELOG-u se na takvo $R(w_1,...,w_m)$ ne može primeniti brisanje iz CWA-pravila, a u DATALOG-u se za takvo $R(w_1,...,w_m)$ ne primenjuje CWA-princip. Oba sistema rade bez korišćenja CWA u skladu sa opštom deduktivnom logikom i dovode do istih rezultata.

Slučaj 3b) $R(w_1,...,w_m)$ se ne može unificirati ni sa jednim literalom iz baze (niti se može pobiti pomoću sopstvenih aksioma, jer bi to bio slučaj pod 1) ili 2)). CWA-pravilo u BASELOG-sistemu briše $R(w_1,...,w_m)$ što odgovara rezoluciji sa $\neg R(w_1,...,w_m)$. Time se usvaja da je $\neg R(w_1,...,w_m)$ dodatna aksioma. Dakle, primena CWA-pravila u BASELOG-u znači prihvatanje da je $\neg R(w_1,...,w_m)$ izvodivo. Tada $R(w_1,...,w_m)$ ne može biti izvodivo ni iz DATALOG programa, pa se po CWA principu u DATALOG-u usvaja tačnost $\neg R(w_1,...,w_m)$. Sledi, ako BASELOG-sistem usvaja $\neg R(w_1,...,w_m)$ onda i DATALOG-sistem usvaja $\neg R(w_1,...,w_m)$.

Obrnuto, ako DATALOG program usvoji $\neg R(w_1,...,w_m)$ po principu CWA, to znači da $R(w_1,...,w_m)$ nije izvodivo, pa se $R(w_1,...,w_m)$ ne nalazi u bazi. Tada unifikacija $R(w_1,...,w_m)$ ni sa jednim literalom iz baze nije moguća, pa se u BASELOG-u primenjuje CWA - pravilo, odnosno usvaja se da je $\neg R(w_1,...,w_m)$ dodatna aksioma u BASELOG-sistemu.

Ovim je dokazano da BASELOG-sistem usvaja $\neg R(w_1,...,w_m)$ akko DATALOG-sistem usvaja $\neg R(w_1,...,w_m)$ (pod pretpostavkom da je R deklarisan kao CWA-predikat).

Sledi, da ako su svi predikati u BASELOG-sistemu uzeti za CWA-predikate, onda BASELOG-sistem radi kao DATALOG.

Time je dokaz završen. □

Primer 4.11. U literaturi [CGT89] se nalazi sledeći primer:
Dat je program Ps gde je simbolom *d* označen EDB predikat, odnosno predikat sadržan u bazi podataka (pojam EDB predikata objašnjen je u odeljku 3.3.1.), simbolima r1,...,r5 označene su sopstvene aksiome, koje se u [CGT89] nazivaju pravilima izvođenja.

r1: p(X,Y) : − ¬q(X,Y), s(X,Y).
r2: q(X,Y) : − q(X,Z), q(Z,Y).
r3: q(X,Y) : − d(X,Y), ¬r(X,Y).
r4: r(X,Y) : − d(Y,X).
r5: s(X,Y) : − q(X,Z), q(Y,T), X≠Y

Elementi baze podataka su:

d(a,b)
d(b,c)
d(e,e)

Kao rezultat rada ovog DATALOG programa u [CGT89] je izvedena činjenica p(b,a).

Pokazaće se kako se ova činjenica može izvesti pomoću BASELOG-sistema.

Polazi se od negacije tvrđenja:

~P(b,a)&

što se priključuje sopstvenim aksiomama.

Ostale sopstvene aksiome su r1,...,r5 iz programa Ps, a bazne aksiome su određene torkama sa EDB predikatima. Oznaka za kraj aksiome je '&' a za negaciju '~', simbol disjunkcije se ne piše.

U internoj formi BASELOG-sistema sopstvene aksiome iz programa Ps imaju sledeći izgled:

AKS[1]='Q(X1,Y1)~S(X1,Y1)P(X1,Y1)&' od r1
AKS[2]='~Q(X1,Z1)~Q(Z1,Y1)Q(X1,Y1)&' od r2
AKS[3]='~D(X1,Y1)R(X1,Y1)Q(X1,Y1)&' od r3
AKS[4]='~D(Y1,X1)R(X1,Y1)&' od r4
AKS[5]='=(X1,Y1)~Q(X1,Z1)~Q(Y1,T1)S(X1,Y1)&' od r5
AKS[6]='~P(b,a)&' od negacije tvrđenja

Bazne aksiome su:

BAKS[1]='D(a,b)&'
BAKS[2]='D(b,c)&'
BAKS[3]='D(e,e)&'

gde su (a,b),(b,c),(e,e) torke (parovi) iz baze, a D je EDB predikat.

Neka je definisana lista CWA-predikata u kojoj se nalaze svi predikati koji učestvuju u programu. Ova lista ima sledeći izgled u BASELOG-sistemu:

CWA[1]='Q'
CWA[2]='R'
CWA[3]='S'
CWA[4]='D'
CWA[5]='P'

To znači da na osnovu ovako definisane liste CWA-predikata (svi predikati koji učestvuju u programu su CWA) BASELOG-sistem radi u režimu zatvorenog sveta odnosno na isti način kao i DATALOG.

BASELOG-sistem generiše prazan sastavak uz sledeće parametre koji definišu okvir pretraživanja:
- maksimalna (granična) dubina pretraživanja: 10 nivoa,
- maksimalna dužina rezolvente: 70 karaktera.

Dokaz ima sledeći izgled:
Uspeh =1
dokaz naden
broj generisanih rezolventi = 20
maksimalno dostignuti nivo = 10
nivo na kojem je generisan prazan sastavak = 10

nivo = 1; centralni sastavak: ~P(b,a)
1. bočni, 3. literal:
Q(X1,Y1)~S(X1,Y1)P(X1,Y1)&

nivo = 2; rezolventa:
/~P(b,a)Q(b,a) ~S(b,a)&
5. bočni, 4. literal:
=(X1,Y1)~Q(X1,Z1)~Q(Y1,T1)S(X1,Y1)&

nivo = 3; rezolventa:
/~P(b,a)Q(b,a)/~S(b,a)=(b,a)~Q(b,Z1)~Q(a,T1)&
3. bočni, 3. literal:
~D(X1,Y1)R(X1,Y1)Q(X1,Y1)&

nivo = 4; rezolventa:
/~P(b,a)Q(b,a) /~S(b,a)=(b,a)~Q(b,Z1) /~Q(a,Y1)~D(a,Y1)R(a,Y1)&
brisanje po CWA-pravilu

nivo = 5; rezolventa:
/~P(b,a)Q(b,a) /~S(b,a)=(b,a)~Q(b,Z1) /~Q(a,Y1)~D(a,Y1)&
8. bočni, 1. literal:
D(a,b)&

nivo = 6; rezolventa:
/~P(b,a)Q(b,a) /~S(b,a)=(b,a)~Q(b,Z1)&
3. bočni, 3. literal:
~D(X1,Y1)R(X1,Y1)Q(X1,Y1)&
nivo = 7; rezolventa:

/~P(b,a)Q(b,a) /~S(b,a)=(b,a)/~Q(b,Y1)~D(b,Y1)R(b,Y1)&
brisanje po CWA-pravilu

nivo = 8; rezolventa:
/~P(b,a)Q(b,a) /~S(b,a)=(b,a)/~Q(b,Y1)~D(b,Y1)&
9. bočni, 1. literal:
D(b,c)&

nivo = 9; rezolventa:
/~P(b,a)Q(b,a)&
brisanje po CWA-pravilu

nivo = 10; rezolventa:
&
DOKAZ JE ZAVRŠEN

Napomena: Na nivou 9 rezolventa je određena primenom tehnike za rad sa jednakošću, koja je sastavni deo ADT sistema (procedura ***ikomb***). Suština te tehnike je u brisanju literala =(t1,t2) kad se termi t1 i t2 ne mogu unificirati, što zamenjuje rezoluciju sa ¬=(t1,t2). Ovu tehniku su razradili P. Hotomski i I. Berković, videti u [LHRB95].

Sledi, BASELOG-sistem je dokazao činjenicu P(b,a).

Ako se parametar maksimalne dužine rezolvente smanji na 65, pri istoj dubini, dobija se identičan dokaz, ali je ukupan broj generisanih rezolventi manji, odnosno 19. Ako se poveća maksimalna dubina sa 10 na 12 nivoa, pri istoj maksimalnoj dužini rezolvente 65 generiše se ukupno veći broj rezolventi, odnosno 37, a dobija se identičan dokaz.
Ishodi rada BASELOG-sistema za različite vrednosti parametara dužine rezolvente i dubinu pretrage u slučaju kada su svi predikati iz programa deklarisani kao CWA-predikati dati su u tabeli 4.5.

CWA-predikati: Q,R,S,D,P	**Broj generisanih rezolventi**	**Maksimalno dostignuti nivo**	**Nivo na kojem je gen. prazan sast.**
Nivo=10; Dužina rezolvente=70 CH	20	10	10
Nivo=10; Dužina rezolvente=65 CH	19	10	10
Nivo=12; Dužina rezolvente=65 CH	37	10	10

Tabela 4.5.
Napomena: oznaka CH u tabeli znači ‘karakter’

To ukazuje da efikasnost BASELOG-sistema u odnosu na obim pretrage bitno zavisi od vrednosti parametara koji određuju maksimalnu dužinu rezolvente i maksimalnu dubinu.

Značajno je uočiti, da BASELOG-sistem dopušta variranje liste CWA-predikata. Tako, ako se u listi CWA navedu samo sledeći predikati:

CWA[1]='Q'
CWA[2]='R'
BASELOG-sistem generiše isti dokaz za parametre pretraživanja dužine 70 i dubine 10, a ukupan broj generisanih rezolventi je 18.
Za maksimalnu dužinu rezolvente 65, pri istoj dubini generiše se identičan dokaz, a ukupan broj generisanih rezolventi je 17.
Za maksimalnu dubinu od 12 nivoa, ukupan broj generisanih rezolventi je 31, dok je dokaz identičan prethodnom.
Ove vrednosti su predstavljene tabelom 4.6.

CWA-predikati: Q i R	Broj generisanih rezolventi	Maksimalno dostignuti nivo	Nivo na kojem je gen. prazan sast.
Nivo=10; Dužina rezolvente=70 CH	18	10	10
Nivo=10; Dužina rezolvente=65 CH	17	10	10
Nivo=12; Dužina rezolvente=65 CH	31	10	10

Tabela 4.6.

Međutim, u slučaju da bar jedan od predikata Q ili R nije deklarisan u listi CWA-predikata ili je lista CWA prazna, BASELOG-sistem ne nalazi dokaz tvrđenja *p(b,a)*, što je u skladu sa očekivanjem.

Primer 4.11.1. Ako se u listi CWA-predikata nalazi samo 'R' rezultat rada BASELOG-sistema u slučaju parametara:
- maksimalna (granična) dubina pretraživanja: 10 nivoa,
- maksimalna dužina rezolvente: 100 karaktera.

uspeh =2
neizvestan ishod dokazivanja
broj generisanih rezolventi = 57
maksimalno dostignuti nivo = 10

Dakle, radeći u sistemu zatvorenog sveta, BASELOG-sistem daje iste rezultate kao DATALOG-sistem. U režimu otvorenog sveta BASELOG-sistem daje korektne rezultate, koji se u DATALOG-u ne mogu dobiti.

Sledeći primer ilustruje situaciju u kojoj DATALOG-sistem ne daje semantički prihvatljiv odgovor, dok BASELOG daje semantički prihvatljiv odgovor.

Primer 4.12. Neka je DATALOG program određen sledećim IDB i EDB predikatima:
IDB predikati u pravilima:

r1: *paralelno*(X,Y):- *paralelno*(Y,X).
r2: *paralelno*(X,Y):- *normalno*(X,Z),*normalno*(Y,Z).

EDB predikati:
paralelno(a,b)
paralelno(b,c)
normalno(a,d)
normalno(b,d)

DATALOG-sistem na osnovu ovog programa ne uspeva da izvede činjenicu *paralelno(a,c)*, pa po CWA principu prihvata da je istina $\neg$*paralelno(a,c)*. To u odnosu na koncept zatvorenog sveta DATALOG-a predstavlja očekivan odgovor, koji međutim, nije u skladu sa stvarnim stanjem. Naime, u programu nedostaje znanje o tranzitivnosti paralelnosti.

Realizacija ovog programa u BASELOG-sistemu nema ovaj nedostatak. U stvari, BASELOG-sistem na upit "paralelno(a,c)", pri čemu *paralelno* nije deklarisan kao CWA-predikat, daje:

uspeh = 0 i nema više rezolventi
neizvestan ishod dokazivanja
broj generisanih rezolventi = 11
maksimalno dostignuti nivo = 6

odn. odgovor BASELOG-sistema u ovom slučaju je "NIJE IZVODIVO" odnosno "NIJE MI POZNATO", što je sa stanovišta obrazovnog računarskog softvera prihvatljivo, dok je odgovor DATALOG-a sa istog stanovišta neprihvatljiv (dezinformišući). □

Upravo to daje prednost BASELOG-u u projektovanju baza podataka u oblasti obrazovnog računarskog softvera. Uzimajući u obzir činjenicu da BASELOG-sistem proširuje mogućnosti DATALOG-a u navedenom smislu, a poklapa se sa njim u radu sa onim bazama podataka, kod kojih je usvojen koncept zatvorenog sveta, uvođenje BASELOG prilaza je ***proširenje*** a ne restrikcija usvojenih koncepata projektovanja baza podataka.

4.6.3. Komparacija BASELOG-a sa PROLOG-om

Rad PROLOG-sistema temelji se na koncepciji KONAČNOG NEUSPEHA:
"Element cilja koji sadrži negaciju, oblika *notC* je zadovoljen ako svaki pokušaj zadovoljenja cilja *C* trpi konačan neuspeh (ne postoji za *C* konačno stablo pretraživanja sa granom uspeha). U suprotnom, ako za *C* postoji konačno stablo pretraživanja sa granom uspeha, onda cilj *notC* trpi konačan neuspeh i ne može se zadovoljiti". [Hot95]

Ovakva koncepcija ne odgovara logičkom tretmanu negacije i dovodi do nekorektnih odgovora PROLOG-sistema. Da bi se oni izbegli uvode se razna ograničenja na upotrebu negacije u PROLOG-u i razne tehnike, koje pomažu, ali ne eliminišu u potpunosti mogućnost generisanja nekorektnih odgovora u vezi sa predikatima koji ne predstavljaju relevantna znanja sadržana u PROLOG programu shvaćenom kao ZATVOREN SVET.

Logički korektan tretman negacije u BASELOG-u eliminiše takve nekorektnosti, ali dopušta "namerni" tretman pojedinih (svih, nekih ili nijednog) predikata u *zatvorenom svetu*. To se može ilustrovati na sledećem primeru.

Primer 4.13. Posmatra se sledeći program:

visok(X): - *not nizak*(X), *covek*(X).
covek(marko).
covek(slavica).
nizak(mirko).

Na pitanje **?- visok(slavica)** PROLOG-sistem daje odgovor **yes**.

Međutim, na pitanje **?- visok(X)** sistem daje odgovor **no**."

U odeljku 3.2.2. je detaljnije objašnjena i strategija konačnog neuspeha u PROLOG-u i dato je tumačenje ovih odgovora.

Pokazaće se kako je ovaj primer realizovan pomoću BASELOG-sistema. Navedeni PROLOG-program u internoj formi BASELOG-sistema ima sledeći izgled:

Sopstvene aksiome:
AKS[1]='VISOK(X1)~COVEK(X1)NIZAK(X1)&"

Bazne aksiome:
BAKS[1]='COVEK(Marko)&'
BAKS[2]='COVEK(Slavica)&'
BAKS[3]='NIZAK(Mirko)&'

Slučaj A. Rad BASELOG-sistema u režimu zatvorenog sveta.

U tom slučaju se u listi CWA-predikata nalaze svi predikati koji učestvuju u programu. Tako lista CWA-predikata ima sledeći izgled:
CWA[1]='VISOK'
CWA[2]='NIZAK'
CWA[3]='COVEK'
Ako se postavi pitanje: Da li je Slavica visoka? odnosno u internoj formi BASELOG-sistema ovo pitanje ima oblik:

~VISOK(Slavica)&

Rezultat rada BASELOG-sistema je:
uspeh = 1
broj generisanih rezolventi = 3
maksimalno dostignuti nivo = 4
nivo na kojem je generisan centralni sastavak = 4

nivo = 1; centralni sastavak: ~VISOK(Slavica)&
1. bočni, 1. literal:
VISOK(X1)~COVEK(X1)NIZAK(X1)&

nivo = 2; rezolventa:
/~VISOK(Slavica)~COVEK(Slavica)NIZAK(Slavica)&
brisanje po CWA-pravilu

nivo = 3; rezolventa:
/~VISOK(Slavica)~COVEK(Slavica)&
5. bočni, 1. literal:
COVEK(Slavica)&

nivo = 4; rezolventa:
&
DOKAZ JE ZAVRŠEN

Znači, BASELOG-sistem je dao isti odgovor kao i PROLOG-sistem: "Da, Slavica je visoka", jer je radio u režimu zatvorenog sveta odnosno kao i PROLOG-sistem.

Dalje, ako se postavi pitanje "Ima li visokih ljudi?" odnosno u internoj formi BASELOG-sistema ovo pitanje ima oblik:

~VISOK(X1)&

BASELOG-sistem će raditi na sledeći način: Generiše rezolventu sa AKS[1]: /~VISOK(X1)~COVEK(X1)NIZAK(X1)&. Kako nema komplementarnog literala sa NIZAK(X1) dalja rezolucija nije moguća pa se uključuje CWA-kontroler. CWA-kontroler utvrđuje da je moguća unifikacija literala NIZAK(X1) sa elementom baze NIZAK(Mirko). Zato se CWA-pravilo ne primenjuje i ne može se generisati nova rezolventa. Dokaz se završava bez generisanja sastavka. BASELOG-sistem daje odgovor:

uspeh = 0
dokaz nije nađen

što znači da "Nema visokih ljudi" i taj odgovor se poklapa sa odgovorom PROLOG-sistema.

Primećuje se da su oba odgovora nekorektna, jer ne slede iz programa. Dobijeni su kao posledica konačnog neuspeha, odnosno, tretmana svih predikata u zatvorenom svetu. Sa stanovišta semantike baze i upita nedopustivo je ove predikate proglasiti za CWA, odnosno treba ih posmatrati u otvorenom svetu. PROLOG to ne omogućava, dok BASELOG dopušta takvu mogućnost. To je sledeći slučaj B.

Slučaj B. Sada će biti pokazano da BASELOG-sistem radi kao univerzalni ADT sistem sa logički korektnim izvođenjem zaključaka ako je lista CWA-predikata prazna, odnosno, BASELOG-sistem radi u režimu otvorenog sveta.

Pod istim uslovima (sopstvene aksiome, bazne aksiome) kao u slučaju A, samo s tom razlikom što je lista CWA-predikata **prazna,** rezultat rada BASELOG-sistema na upit ~VISOK(Slavica)& je:

Nemogućnost generisanja novih rezolventi nakon generisanja rezolvente /~VISOK(Slavica)~COVEK(Slavica)NIZAK(Slavica)&, jer NIZAK nije CWA-predikat.

Zato sistem saopštava odgovor:
uspeh = 0
dokaz nije nađen
broj generisanih rezolventi = 1
maksimalno dostignuti nivo = 2

koji se u skladu sa tabelom 4.4. tumači kao "NIJE IZVODIVO". To znači da se na osnovu datih činjenica i pravila ne može zaključiti da je Slavica visoka.

Na postavljeno pitanje "Ima li visokih ljudi?" odnosno u internoj formi BASELOG-sistema:

~VISOK(X1)&

iz istih razloga sistem, takođe, daje odgovor "NIJE IZVODIVO" što je korektno, jer se u bazi ne pominju visoki ljudi.

PROLOG-sistem omogućava nalaženje više različitih načina zadovoljenja postavljenog cilja i dobijanje odgovora na upit, koji nisu u potpunosti instancirani konstantama. BASELOG-sistem, takođe, omogućava dobijanje više rešenja. To ilustruje sledeći primer:

Primer 4.13.1. Data je sopstvena aksioma:

AKS[1]='VISOK(X1)~COVEK(X1)NIZAK(X1)&"

Bazne aksiome:
BAKS[1]='COVEK(Marko)&'
BAKS[2]='COVEK(Slavica)&'
BAKS[3]='NIZAK(Mirko)&'
BAKS[4]='VISOK(Jova)&'
BAKS[5]='VISOK(Mira)&'

Na postavljeno pitanje "Ima li visokih ljudi?" odnosno u internoj formi BASELOG-a:

~VISOK(X1)&

sistem daje odgovore:

zamena: Jova/X1*

zamena: Mira/X1*

pri čemu izraz "Jova/X1" označava da se promenljiva X1 instancira sa termom "Jova", a simbolom "*" označava se kraj zamene.

O načinima na osnovu kojih se izračunava odgovor, odnosno, dolazi do više različitih načina zadovoljenja cilja detaljnije je opisano u [Ber97] i [Hot95].

4.7. Informacije o programskoj realizaciji BASELOG-sistema i mogućnostima proširenja

BASELOG-sistem je implementiran na personalnom računaru i radi pod MS-DOS operativnim sistemom. Implementacija je izvršena na računaru klase IBM PC 486. Realizacija je izvršena na programskom jeziku Pascal (Turbo Pascal ver. 6.0 kompanije Borland International Inc.). Baza podataka je kreirana upotrebom softvera za baze podataka FoxPro ver. 2.6 za operativni sistem DOS kompanije Ashton Tate. Podaci su memorisani u datoteci koja je DBF formata.

S obzirom da je BASELOG-sistem proširenje u odnosu na ADT sistem [Ber94], potrebno je reći da se ADT sistem zasniva na lineranoj uređenoj OL rezoluciji sa markiranim literalima. Projektovani sistem ADT je sistem sa varijabilnim strategijama pretraživanja i on raspolaže sa tri sintaksne strategije pretraživanja: strategija pregleda u širinu, strategija pregleda u dubinu i kombinovana strategija. U [Ber94] je pokazano da je dubinska strategija najefikasnija, pa je u razvoju BASELOG-sistema upotrebljen ADT sistem koji koristi dubinsku strategiju.

Kao što je rečeno u odeljku 4.4., BASELOG-sistem je proširenje u odnosu na ADT sistem putem liste CWA-predikata, CWA-kontrolera i aksioma tipa BAKS. Ove komponente BASELOG-sistema programski su implementirane na sledeći način:

- **Lista CWA-predikata** realizovana je u BASELOG-sistemu putem niza **CWA**. Niz **CWA** ima dužinu od **brcwa** članova i dužinu ovog niza može da definiše korisnik.

- **CWA-kontroler** je realizovan putem procedure **cwa_kontroler** čiji se listing nalazi u prilogu. Ova procedura se poziva u proceduri **idubrez.** Lista parametara procedure **idubrez** je proširena, zbog implementacije **CWA-kontrolera,** sledećim parametrima:
  1. aks, baks, cwa parametri koji su tipa **akss.** (Tip **akss** je niz stringova čija je dužina jednaka maksimalnom broju aksioma u sistemu)
  2. sn i **bn** - broj sopstvenih i baznih aksioma,
  3. brcwa - broj CWA-predikata u listi CWA
  4. cwa_dn (lokalna promenljiva procedure **idubrez**). Ova promenljiva je tipa niz, čiji su elementi celi brojevi. Maksimalna dužina ovog niza je jednaka maksimalnom broju nivoa pretrage. Elementi ovog niza dobijaju vrednost 1, ako je CWA-pravilo bilo primenjeno na datom nivou pretrage, u suprotnom dobijaju vrednost 0.

U proceduri **cwa_kontroler** poziva se procedura **iunifi**. Ova procedura ima izlazni parametar - promenljivu ***uspeo_uni,*** čija je vrednost 0, ako pokušaj unifikacije poslednjeg literala date rezolvente sa svakom aksiomom tipa BAKS nije uspeo.

Ako pokušaj unifikacije nije uspeo, pozivaju se procedure **isaz, iskr** i **ibris** za realizaciju operacije brisanja poslednjeg literala iz rezolvente i operacije sažimanja i skraćivanja.

- **Aksiome tipa BAKS** su definisane putem niza tipa **akss** čiju dužinu određuje korisnik. Elementi ovog niza su tipa **string** i dužinu tih elemenata, takođe, određuje korisnik. Aksiome tipa BAKS moguće je formirati ili unošenjem putem tastature ili putem postupka za formiranje predikata (aksioma) na osnovu podataka u DBF formatu, što je opisano u poglavlju 5. Realizovano je učitavanje cele baze podataka u operativnu memoriju.

Mogućnosti proširenja BASELOG-sistema su:

- razvoj korisničkog interfejsa za davanje semantičkih odgovora koji će omogućiti korisniku da koristi sistem bez potrebe za poznavanjem unutrašnjih mehanizama BASELOG-sistema. Kao što je objašnjeno u odeljku 4.5.1. taj interfejs treba da obezbedi povezivanje internih odgovora u sistemu sa semantičkim odgovorima koji se saopštavaju korisniku,

- putem posebnih interfejsa može se obezbediti uključivanje samo aktuelnog segmenta realne baze u režim rada BASELOG-sistema,

- razvoj interfejsa za uključivanje realne baze podataka koja se nalazi i u drugim formatima kao što su: ORACLE, Access (trenutno je razvijen interfejs za učitavanje podataka iz DBF formata).

5. PROJEKTOVANJE BAZA PODATAKA U OBRAZOVNOM RAČUNARSKOM SOFTVERU POMOĆU BASELOG SISTEMA

5.1. Opšti koncept obrazovnog računarskog softvera koji koristi deduktivne baze zasnovane na BASELOG-sistemu

Opšti koncept projektovanja baza podataka u oblasti obrazovnog softvera mogao bi da se definiše sažeto na sledeći način: s obzirom da se baze podataka u obrazovanju posmatraju zavisno od sadržaja kao otvoren svet ili kao zatvoren svet, potrebno ih je projektovati tako da se uključe ove mogućnosti.

Osnovna karakteristika BASELOG-sistema je integracija koncepata otvorenog i zatvorenog sveta u jedinstven sistem. Izborom liste CWA-predikata, projektant nastavne baze podataka upravlja ponašanjem BASELOG-sistema i omogućuje dobijanje preciznih i korektnih odgovora i eliminiše mogućnost davanja nedopustivih odgovora. Zato se procedura, kojom je realizovan ovaj koncept i zove **CWA-*kontroler***.

BASELOG-sistem se može koristiti u nastavi na sledeće načine:

- kao upitni jezik nad bazom podataka. Oblici upita mogu biti:
  1) Da li je Slavica visoka? (?- visok(slavica))
  Odgovori BASELOG-sistema na upite ovog oblika su: Da, Ne, Neizvesno.
  2) Odgovori na ciljeve koji nisu u potpunosti instancirani konstantama, odnosno dobijanje više mogućih odgovora na upit:
  2.a Lista visokih ljudi: (?- visok(X))
  2.b Prikaži imena i prezimena studenata iz Zrenjanina koji su na smeru *informatika*: (izaberi(X1,X2):- student(X1,X2, Zrenjanin, informatika))
  2.c Prikaži dela Laze Lazarevića: (?- delo(Laza, Lazarevic, X1))

- upotrebom sistema automatskog rezonovanja (BASELOG-sistema) može se dedukovati činjenica koja se ne nalazi u bazi podataka, odnosno, u odnosu na klasične baze podataka pristup upotrebom ovakvih sistema znači mogućnost dobijanja odgovora na upite, čije se komponente ne nalaze u bazi podataka, već se odgovor dedukuje na osnovu činjenica i znanja, automatskim logičkim rasuđivanjem.

- kao sistem koji pokušava da izvede protivrečnost odnosno da dokaže da negacija izjave logički sledi iz kreirane baze znanja i kao sistem za proveru učeničkih hipoteza

(primeri se nalaze u **odeljku 2.3.2. *Provera učeničkih hipoteza u dijaloškim sistemima***).

Određivanje CWA-predikata u projektovanju obrazovnog softvera

Određivanje CWA-predikata može se opisati na sledeći način:

- Ako je svojstvo potpuno opisano u bazi, odnosno, baza podataka je semantički kompletna u odnosu na predikat koji odgovara tom svojstvu, onda se takav predikat uzima za CWA-predikat.

- U suprotnom, kada projektant baze nije siguran da je u bazi odrazio sve relevantne aspekte nekog svojstva, onda odgovarajući predikat ne sme da bude deklarisan kao CWA-predikat.

Sažeto rečeno, princip izbora CWA-predikata glasi:

Predikat se deklariše kao CWA-predikat akko je baza kompletna u odnosu na svojstvo koje taj predikat opisuje.

CWA-predikat je onaj predikat iz baze, za koga je potrebno da sistem radi u režimu zatvorenog sveta. To je slučaj kod testiranja (predikati koji su rešenja) ili kod definicija pojmova (baza podataka koja sadrži definicije pojmova). Na primer: baza podataka sadrži informacije o jugoslovenskim rekama (predikat JUGOSL_REKA), ili, baza podataka sadrži definicije osnovnih geometrijskih figura (predikat TROUGAO, KVADRAT), ili baza podataka sadrži informacije o američkim piscima (predikat AMER_PISAC).

Za predikate koji su u listi CWA BASELOG-sistem radi u režimu DATALOG-a (zatvoreni svet). Za sve ostale predikate BASELOG-sistem radi u režimu sistema ADT (otvoreni svet), pri čemu je sistem oslobođen nedostatka PROLOG-a u odnosu na tretman negacije i koncepciju konačnog neuspeha.

Prilikom određivanja CWA-predikata mora se precizno znati semantika predikata. Projektant nastavne baze podataka mora dobro da zna u kom obliku i u kom obimu se nalaze podaci u bazi (da li svi podaci o krugu, da li su podaci o zbiru uglova u trouglu u stepenima ili radijanima).

Predikati koji se koriste u testiranju moraju biti CWA-predikati. Na primer, u slučaju da se postavi pitanje učeniku: Da li je kvadrat paralelogram? upitni predikat *kvadrat* mora biti CWA-predikat (odgovor mora biti tipa: DA ili NE).

Jedan od mogućih pristupa u projektovanju baze podataka upotrebom BASELOG-sistema se ogleda u sledećem - kreirati bazu podataka upotrebom nekog od softvera za rukovanje

bazama podataka, kako bi se iskoristili mehanizmi koje ima SRBP za definisanje integriteta baze podataka:

- definisanje domena obeležja,
- definisanje integritetne komponente (definisanje primarnog ključa šeme relacije a samim tim i funkcionalnih zavisnosti kao najvažnijeg ograničenja integritetne komponente),
- definisanje referencijalnih integriteta

a zatim tako projektovanu i kreiranu bazu podataka predstaviti putem aksioma tipa BAKS u BASELOG-sistemu. Pri tom, iskoristiti mogućnosti BASELOG-sistema:

- logički korektan mehanizam zaključivanja,
- mogućnost izbora režima rada: otvoren, zatvoren, poluotvoren svet.

Ovakav pristup zahteva razvoj i implementaciju procedura, koje od postojećeg formata zapisa podataka iz konkretnog softvera za rukovanje bazama podataka kreiraju predikate (aksiome tipa BAKS). Trenutno je to rešeno za DBF format zapisa podataka koga kreiraju dBase softveri i Foxpro softveri za personalne računare.

U postupku povezivanja BASELOG-sistema sa konkretnom bazom podataka postoje dve mogućnosti:

- BASELOG-sistem se povezuje sa relacionom bazom podataka (skup relacija u kojima su obeležja standardni tipovi podataka)

- BASELOG-sistem se povezuje sa *predikatskom* bazom podataka. U pitanju je, takođe, relaciona baza podataka ali su vrednosti obeležja u relacijama već formirani predikati.

Postupci za povezivanje BASELOG-sistema sa konkretnim bazama podataka objašnjeni su detaljnije u poglavlju 5.2.

U daljem razvoju BASELOG-sistema moguće je rešiti sledeće:

- povezivanje BASELOG-sistema sa bazama podataka Windows okruženja putem ODBC protokola, kako bi se omogućio pristup BASELOG-a različitim formatima podataka (formati Access-a, ORACLE-a, SQL3, itd.)

- ekstrahovanje samo željenih torki iz baze (prvo se vrši pretraživanje po bazi, pa se na osnovu tako selektovanog dela formiraju BAKS aksiome). Pretraživanje može da se realizuje u posebnoj proceduri BASELOG-sistema, pre nego što se formiraju aksioma

tipa BAKS (pre poziva procedure DBFUBaks odnosno PDBFUBaks, koje su detaljnije objašnjene u odeljku 5.2.).

- Sam BASELOG-sistem na osnovu datih kriterijuma određuje podatke iz baze podataka na osnovu kojih će se formirati aksiome tipa BAKS.

Projektovanje nastavnih baza podataka upotrebom BASELOG-sistema ima za cilj da smanji neodređenost odgovora sistema (PROLOG, ADT) i da eliminiše mogućnost davanja netačnih odgovora (DATALOG, kao i klasični sistemi za baze podataka koji, takođe, rade po principu zatvorenog sveta.).

5.2. Povezivanje BASELOG-sistema sa konkretnom bazom

Elemente za rad BASELOG-sistema:

- upit,
- sopstvene aksiome,
- bazne aksiome i
- listu CWA-predikata

moguće je učitati iz postojeće datoteke na disku (iz datoteke **AKSIOME.TXT**) ili je na početku rada moguće kreirati nove podatke. Novi podaci se, takođe, upisuju u fajl AKSIOME.TXT.

Za potrebe povezivanja BASELOG-a sa konkretnom bazom podataka kreirana je ***programska jedinica*** na programskom jeziku Pascal (Turbo Pascal-u ver. 7.0 kompanije Borland International Inc.). Ova jedinica se zove **DBASE** i koristi se u procedurama **DBFUBaks** i **PDBFUBaks**, koje su sastavni delovi BASELOG-sistema. Listinzi procedura **DBFUBaks** i **PDBFUBaks** nalaze se u prilogu 4 i prilogu 5.

U programskoj jedinici DBASE kreirane su osnovne funkcije za rad datotekom:

- učitavanje datoteke sa diska u operativnu memoriju,
- određivanje broja slogova u datoteci,
- funkcije za rad sa stringovima (odsecanje praznina sa desne, leve i sa obe strane stringa),
- dodavanje novog sloga u datoteku,
- brisanje sloga iz datoteke,
- brisanje sadržaja cele datoteke,
- zatvaranje datoteke.

Procedure **DBFUBaks** i **PDBFUBaks** formiraju predikate (aksiome tipa BAKS) na osnovu podataka u već formiranoj datoteci koja je DBF formata (ovaj format datoteke se kreira putem softvera za baze podataka za personalne računare: FoxPro ili dBASE).

Obeležja datoteke mogu uzimati vrednosti iz raznih domena, kao na primer: skup imena država, skup imena gradova, datumi, godine studija, ili domeni mogu biti skup već formiranih predikata. Ako se radi o bazi podataka, čiji su domeni obeležja standardne vrednosti (nizovi znakova, brojevi, datumi), onda aksiome tipa BAKS formira procedura **DBFUBaks.** Ako su domeni obeležja skup već formiranih predikata, onda aksiome tipa BAKS formira procedura **PDBFUBaks**.

Povezivanje BASELOG-sistema sa relacionom bazom podataka

Za slučaj da su vrednosti obeležja standardnog tipa: nizovi karaktera, brojevi, datumi, aksiome tipa BAKS se formiraju na sledeći način.

Predikat (aksioma tipa BAKS) ima isto ime kao i naziv datoteke formata DBF. Ove aksiome se upisuju u fajl AKSIOME.TXT. (Moguće je da se aksiome ne zapisuju u datoteku, nego se nalaze samo u operativnoj memoriji.) Argumenti predikata dobijaju redom vrednosti koje su jednake vrednostima obeležja u datoteci.

Primer 5.1. Kreirana je datoteka STUDENT.DBF sa sledećom logičkom strukturom obeležja:

(IME, PREZIME, GOD_STUD, BROJ_INDEKSA)

i neka se u njoj nalazi sledeći slog (torka):

(Mirko, Ilic, 2, 23/92-021) (Slika 5.1.)

STUDENT.DBF

IME	PREZIME	GOD_STUD	BROJ_INDEKSA
Mirko	Ilic	2	23/92-021
.....	.....	.....	.....

Slika 5.1.

Na osnovu ovog sloga formira se sledeća aksioma tipa BAKS BASELOG-sistema:

BAKS[1]:='STUDENT(Mirko,Ilic,2,23/92-021)&'

To znači da se i u BASELOG-sistemu povezivanje predikata (aksioma tipa BAKS) sa relacijom u bazi podataka (datotekom formata DBF) vrši na isti način kao i u DATALOG-u u slučaju povezivanja EDB predikata sa konkretnom relacijom. U DATALOG-u ime EDB predikata je ime relacije, dok se argumenti predikata povezuju sa obeležjima relacije u odnosu na svoju poziciju u predikatu.

Prilikom definisanja sopstvenih aksioma treba voditi računa da naziv u literalu, koji se povezuje sa predikatom iz baze, bude identičan nazivu u aksiomi tipa BAKS.

Primer 5.2. Definisana je sledeća sopstvena aksioma:

AKS[1]:='~PARALELNO(X,Y)PARALELNO(Y,X)&'

odnosno u implikativnoj formi ona ima oblik:

$$paralelno(x,y) \Rightarrow paralelno(y,x)$$

Podaci se nalaze u datoteci PARALELNO.DBF i na osnovu ove datoteke formiran je niz BAKS aksioma:

BAKS[1]:='PARALELNO(a,b)&'
BAKS[2]:='PARALELNO(b,c)&'

U fajlu AKSIOME.TXT, osim baznih aksioma nalaze se i sledeći elementi:

- negacija teoreme (upit, odnosno niz KN[i], $i=1,...m$),
- sopstvene aksiome (niz AKS[i], $i=1,...sn$) i
- lista CWA-predikata (niz CWA[i], $i=1,...,brcwa$)

pri čemu važi:

$$n = m + sn + bn$$

gde je ***n*** ukupan broj aksioma BASELOG-sistema, a ***bn*** ukupan broj BAKS aksioma.

Ograničenja prvobitno implementiranog BASELOG-sistema u odnosu na povezivanje sa konkretnom bazom podataka su sledeća:

- tip podataka za predstavljanje aksioma je niz. Dužina tog niza (promenljiva **akss** BASELOG-sistema) u kome se smeštaju sopstvene i bazne aksiome je limitirana, zbog toga što je u pitanju statička alokacija memorije (niz). To znači da je ukupan broj sopstvenih aksioma i baznih aksioma (torki iz relacione baze podataka) limitiran.

- dužina rezolvente. Ovaj parametar može da menja korisnik ako je to potrebno, u odnosu na dužinu baznih aksioma (ukupan zbir dužine obeležja torke iz baze podataka).

U daljem razvoju BASELOG-sistema realizovano je predstavljanje podataka (sopstvenih i baznih aksioma i liste CWA-predikata) putem dinamičke alokacije memorije (niz pokazivača na tip podataka string), čime je omogućeno učitavanje većeg obima podataka iz datoteke.

Povezivanje BASELOG-sistema sa bazom podataka u kojoj su vrednosti obeležja već formirani predikati

Za potrebe povezivanja BASELOG-sistema za bazom podataka u kojoj su vrednosti obeležja već formirani predikati razvijena je procedura **PDBFUBaks.** Parametri ove procedure su: **imedbf** i **obelezje**, na osnovu kojih korisnik određuje ime konkretne datoteke i naziv njenog obeležja na osnovu koga će se formirati aksiome BAKS.

Primer 5.3. Kreirana je datoteka PODACI.DBF koja ima samo jedno obeležje PREDIKAT, *dom*(PREDIKAT)=chr(200). Konkretne vrednosti su:

PREDIKAT
JE_MEDITERAN_DRŽAVA(Portugalija)
.....
IMA_IZLAZ_NA_MORE(Engleska)
.....
GL_GRAD_MEDIT_DRŽ(Lisabon,Portugalija)
.....
JE_GLAVNI_GRAD(London,Engleska)
.....

Ako prilikom startovanja BASELOG-sistema korisnik unese naziv datoteke: PODACI (ekstenzija nije potrebna) i naziv obeležja: PREDIKAT, onda će procedura **PDBFUBaks** na osnovu ovih podataka, kreirati sledeće aksiome tipa BAKS:
BAKS[1]:='JE_MEDITERAN_DRŽAVA(Portugalija)&'
.....
BAKS[s]:='IMA_IZLAZ_NA_MORE(Engleska)&'
.....
BAKS[t]:='GL_GRAD_MEDIT_DRŽ(Lisabon,Portugalija)&'
.....
BAKS[u]:='JE_GLAVNI_GRAD(London,Engleska)&'
.....
pri čemu je 1< s < t < u < z, gde je ***z*** ukupan broj slogova u datoteci PODACI.DBF.

5.3. Primer rada BASELOG-sistema na realnoj bazi znanja

U ovom odeljku navodi se nekoliko primera rada BASELOG-sistema na realnoj bazi znanja, pri čemu prvo slede primeri sa predikatima jednog tipa, a potom slede primeri sa predikatima različitog tipa.

Primer 5.4. **GEOGRAFIJA.** U bazi podataka se nalaze sledeći elementi:
JE_MEDITERAN_DRŽAVA(Portugalija)
JE_MEDITERAN_DRŽAVA(Španija)
JE_MEDITERAN_DRŽAVA(Francuska)
JE_MEDITERAN_DRŽAVA(Italija)
JE_MEDITERAN_DRŽAVA(Slovenija)
JE_MEDITERAN_DRŽAVA(Hrvatska)
JE_MEDITERAN_DRŽAVA(Jugoslavija)
JE_MEDITERAN_DRŽAVA(Grčka)
JE_MEDITERAN_DRŽAVA(Turska)
JE_MEDITERAN_DRŽAVA(Sirija)
JE_MEDITERAN_DRŽAVA(Jordan)
JE_MEDITERAN_DRŽAVA(Liban)
JE_MEDITERAN_DRŽAVA(Izrael)
JE_MEDITERAN_DRŽAVA(Egipat)
JE_MEDITERAN_DRŽAVA(Libija)
JE_MEDITERAN_DRŽAVA(Alžir)
JE_MEDITERAN_DRŽAVA(Tunis)
JE_MEDITERAN_DRŽAVA(Maroko)

Predikat JE_MEDITERAN_DRŽAVA može sa sigurnošću da se proglasi za CWA-predikat, jer se polazi od pretpostavke da se imena ***svih*** mediteranskih država nalaze u bazi. Ova baza sadrži ***kompletno*** znanje o svojstvu ***mediteranska država***.

Na primer, na upit: Da li je Bugarska mediteranska država?, odnosno, u formi BASELOG-a:

~JE_MEDITERAN_DRŽAVA(Bugarska)&

odgovor BASELOG-sistema je NE ako se baza podataka posmatra kao zatvoren svet, odnosno ako se predikat JE_MEDITERAN_DRŽAVA proglasi za CWA-predikat.

NAPOMENA: Pitanje ***ažuriranja*** podataka u bazi ne treba mešati sa pojmom ***kompletnosti*** baze. Kompletnost se posmatra pod pretpostavkom da je baza ažurirana. Ako baza nije ažurirana, onda pitanje njene kompletnosti gubi smisao.

Primer 5.5. **GEOGRAFIJA.** U bazi podataka se nalaze sledeći elementi:
IMA_IZLAZ_NA_MORE(Engleska)
IMA_IZLAZ_NA_MORE(Jugoslavija)
IMA_IZLAZ_NA_MORE(Francuska)
IMA_IZLAZ_NA_MORE(Španija)
IMA_IZLAZ_NA_MORE(Australija)
IMA_IZLAZ_NA_MORE(Angola)
IMA_IZLAZ_NA_MORE(SAD)
IMA_IZLAZ_NA_MORE(Indija)
IMA_IZLAZ_NA_MORE(Kina)
IMA_IZLAZ_NA_MORE(Južna_Koreja)
IMA_IZLAZ_NA_MORE(Kanada)
IMA_IZLAZ_NA_MORE(Libija)
IMA_IZLAZ_NA_MORE(Saudijska_Arabija)
IMA_IZLAZ_NA_MORE(Argentina)
IMA_IZLAZ_NA_MORE(Čile)
IMA_IZLAZ_NA_MORE(Švedska)
IMA_IZLAZ_NA_MORE(Nemačka)

Ova baza nije kompletna u odnosu na predikat IMA_IZLAZ_NA_MORE, jer ne sadrži imena ***svih*** zemalja koje imaju more.

Na primer, na upit: Da li Grčka ima izlaz na more?, odnosno u formi BASELOG-a:

~IMA_IZLAZ_NA_MORE(Grčka)&

u slučaju da se "IMA_IZLAZ_NA_MORE" deklariše za CWA-predikat, odgovor BASELOG-sistema je NE (činjenica IMA_IZLAZ_NA_MORE(Grčka) ne postoji u bazi) jer se baza podataka posmatra kao zatvoren svet. Zato se ova baza mora posmatrati kao ***otvoren*** svet, odnosno, predikat IMA_IZLAZ_NA_MORE se ne sme proglasiti za CWA-predikat, jer ova baza nije semantički kompletna.

Primer 5.6. **GEOGRAFIJA**. Neka se u bazi podataka nalazi predikat JE_GLAVNI_GRAD(X,Y) sa značenjem da je X glavni grad države Y.

JE_GLAVNI_GRAD(London,Engleska)
JE_GLAVNI_GRAD(Pariz,Francuska)
JE_GLAVNI_GRAD(Bon,Nemačka)
JE_GLAVNI_GRAD(Santjago,Čile)
JE_GLAVNI_GRAD(Tokio,Japan)
JE_GLAVNI_GRAD(Beč,Austrija)
JE_GLAVNI_GRAD(Brazilija,Brazil)
JE_GLAVNI_GRAD(Karakas,Venecuela)

JE_GLAVNI_GRAD(Kairo,Egipat)
JE_GLAVNI_GRAD(Atina,Grčka)

Predikat JE_GLAVNI_GRAD se ne može proglasiti za CWA-predikat, jer je pretpostavka da u bazi podataka nisu pohranjene informacije o glavnim gradovima ***svih*** država sveta.

Na primer, na učenikovo pitanje:

Da li je Njujork glavni grad SAD?

u slučaju kada bi se baza podataka posmatrala kao otvoren svet, odgovor BASELOG-sistema bio bi NIJE IZVODIVO (ne znam), što znači da je prisutna neodređenost u odgovoru.

Kada bi se ista baza podataka posmatrala kao zatvoren svet, onda bi odgovor sistema bio NE (što je određen i tačan odgovor i kao takav pedagoški prihvatljiv).

Međutim, s obzirom na to da baza podataka nije kompletna u odnosu na svojstvo ***glavni grad***, ne bi bilo dobro predikat JE_GLAVNI_GRAD proglasiti za CWA-predikat. Tako, na primer, na učenikovo pitanje:

Da li je Otava glavni grad Kanade?

u slučaju da se baza podataka posmatra kao zatvoren svet i nema te činjenice u bazi podataka, odgovor sistema bi bio NE, što je netačno i dezinformiše učenika. Zbog toga, se u takvim slučajevima uzima koncept otvorenog sveta (bolje je da odgovori budu neodređeni nego netačni).

Primer 5.7. **ASTRONOMIJA.** U bazi podataka se nalaze sledeći elementi:
PLANETA_SUNČ_SISTEMA(Merkur)
PLANETA_SUNČ_SISTEMA(Venera)
PLANETA_SUNČ_SISTEMA(Zemlja)
PLANETA_SUNČ_SISTEMA(Mars)
PLANETA_SUNČ_SISTEMA(Jupiter)
PLANETA_SUNČ_SISTEMA(Saturn)
PLANETA_SUNČ_SISTEMA(Uran)
PLANETA_SUNČ_SISTEMA(Neptun)
PLANETA_SUNČ_SISTEMA(Pluton)

Predikat PLANETA_SUNČ_SISTEMA se, u nastavnom smislu, sa sigurnošću može proglasiti za CWA-predikat. Tako na primer, na učenikovo pitanje: Da li je Ganimed planeta sunčevog sistema?, odnosno u formi BASELOG-a:

~PLANETA_SUNČ_SISTEMA(Ganimed)&

sistem mora da odgovori sa NE (Ganimed je satelit Jupitera) odnosno sistem treba da radi u režimu zatvorenog sveta.

Primer 5.8. **HEMIJA.** U bazi podataka se nalaze sledeći elementi:
JE_NEMETAL(vodonik)
JE_NEMETAL(fluor)
JE_NEMETAL(hlor)
JE_NEMETAL(brom)
JE_NEMETAL(jod)
JE_NEMETAL(kiseonik)
JE_NEMETAL(sumpor)
JE_NEMETAL(selen)
JE_NEMETAL(telur)
JE_NEMETAL(azot)
JE_NEMETAL(helijum)
JE_NEMETAL(neon)
JE_NEMETAL(argon)
JE_NEMETAL(kripton)
JE_NEMETAL(ksenon)
JE_NEMETAL(fosfor)
JE_NEMETAL(arsen)
JE_NEMETAL(antimon)
JE_NEMETAL(ugljenik)
JE_NEMETAL(silicijum)
JE_NEMETAL(bor)

Predikat JE_NEMETAL se sa sigurnošću može proglasiti za CWA-predikat, jer je pretpostavka da se u bazi podataka nalaze imena ***svih*** nemetala.

Na primer, za učenikovu hipotezu "Živa je nemetal", odnosno u formi BASELOG-a:

~JE_NEMETAL(ziva)&

sistem treba da otkrije protivrečnost u hipotezi učenika, odnosno da odgovori sa NE, što znači: živa nije nemetal. Deklarisanjem JE_NEMETAL za CWA-predikat obezbeđuje se upravo takav rad BASELOG-sistema.

Primer 5.9. **SLIKARSTVO.** U bazi podataka se nalaze podaci o vodećim impresionistima (podaci su preuzeti iz [OL71]):

IMPRESIONISTA(Mone)
IMPRESIONISTA(Pisaro)
IMPRESIONISTA(Dega)
IMPRESIONISTA(Sisle)
IMPRESIONISTA(Renoar)
IMPRESIONISTA(Sezan)
IMPRESIONISTA(Gijomen)
IMPRESIONISTA(Buden)
IMPRESIONISTA(Mane)
IMPRESIONISTA(Morizo)
IMPRESIONISTA(Kajbot)

Ako učenik postavi pitanje: Da li je Van Gog impresionista? što u internoj formi BASELOG-a ima oblik:

~IMPRESIONISTA(Van_Gog)&

sistem mora da da odgovor NE (Van Gog je ekspresionista), jer se pretpostavlja da su u bazi podataka informacije o svim vodećim slikarima - impresionistima, pa se predikat IMPRESIONISTA proglašava za CWA-predikat.

U prethodnim primerima su posmatrane različite baze podataka, koje sadrže jedan predikat. U stvarnosti, baze sadrže više različitih predikata. Neki od njih treba da budu CWA-predikati, a drugi ne. Deklarisanje koji će predikat biti CWA-predikat zavisi od sadržaja baze, odnosno, njene kompletnosti u odnosu na taj predikat (svojstvo koje on opisuje). To ilustruje sledeći kompleksniji primer, koji integriše koncepte otvorenog i zatvorenog sveta u BASELOG-sistemu.

Primer 5.10. U bazi podataka se nalaze sledeći elementi:
(CWA jednomesni)
JE_MEDITERAN_DRŽAVA(Portugalija)
JE_MEDITERAN_DRŽAVA(Španija)
JE_MEDITERAN_DRŽAVA(Francuska)
JE_MEDITERAN_DRŽAVA(Italija)
JE_MEDITERAN_DRŽAVA(Slovenija)
JE_MEDITERAN_DRŽAVA(Hrvatska)
JE_MEDITERAN_DRŽAVA(Jugoslavija)
JE_MEDITERAN_DRŽAVA(Grčka)
JE_MEDITERAN_DRŽAVA(Turska)
JE_MEDITERAN_DRŽAVA(Sirija)
JE_MEDITERAN_DRŽAVA(Jordan)
JE_MEDITERAN_DRŽAVA(Liban)
JE_MEDITERAN_DRŽAVA(Izrael)
JE_MEDITERAN_DRŽAVA(Egipat)

JE_MEDITERAN_DRŽAVA(Libija)
JE_MEDITERAN_DRŽAVA(Alžir)
JE_MEDITERAN_DRŽAVA(Tunis)
JE_MEDITERAN_DRŽAVA(Maroko)

(O(pen) W(orld) jednomesni)
IMA_IZLAZ_NA_MORE(Engleska)
IMA_IZLAZ_NA_MORE(Jugoslavija)
IMA_IZLAZ_NA_MORE(Francuska)
IMA_IZLAZ_NA_MORE(Španija)
IMA_IZLAZ_NA_MORE(Australija)
IMA_IZLAZ_NA_MORE(Angola)
IMA_IZLAZ_NA_MORE(SAD)
IMA_IZLAZ_NA_MORE(Indija)
IMA_IZLAZ_NA_MORE(Kina)
IMA_IZLAZ_NA_MORE(Južna_Koreja)
IMA_IZLAZ_NA_MORE(Kanada)
IMA_IZLAZ_NA_MORE(Libija)
IMA_IZLAZ_NA_MORE(Saudijska_Arabija)
IMA_IZLAZ_NA_MORE(Argentina)
IMA_IZLAZ_NA_MORE(Čile)
IMA_IZLAZ_NA_MORE(Švedska)
IMA_IZLAZ_NA_MORE(Nemačka)

(CWA dvomesni)
GL_GRAD_MEDIT_DRŽ(Lisabon,Portugalija)
GL_GRAD_MEDIT_DRŽ(Madrid,Španija)
GL_GRAD_MEDIT_DRŽ(Pariz,Francuska)
GL_GRAD_MEDIT_DRŽ(Rim,Italija)
GL_GRAD_MEDIT_DRŽ(Ljubljana,Slovenija)
GL_GRAD_MEDIT_DRŽ(Zagreb,Hrvatska)
GL_GRAD_MEDIT_DRŽ(Beograd,Jugoslavija)
GL_GRAD_MEDIT_DRŽ(Atina,Grčka)
GL_GRAD_MEDIT_DRŽ(Ankara,Turska)
GL_GRAD_MEDIT_DRŽ(Damask,Sirija)
GL_GRAD_MEDIT_DRŽ(Aman,Jordan)
GL_GRAD_MEDIT_DRŽ(Bejrut,Liban)
GL_GRAD_MEDIT_DRŽ(Jerusalim,Izrael)
GL_GRAD_MEDIT_DRŽ(Kairo,Egipat)
GL_GRAD_MEDIT_DRŽ(Tripoli,Libija)
GL_GRAD_MEDIT_DRŽ(Alžir,Alžir)
GL_GRAD_MEDIT_DRŽ(Tunis,Tunis)
GL_GRAD_MEDIT_DRŽ(Rabat,Maroko)

(O(pen) W(orld) dvomesni)
JE_GLAVNI_GRAD(London,Engleska)
JE_GLAVNI_GRAD(Pariz,Francuska)
JE_GLAVNI_GRAD(Bon,Nemačka)
JE_GLAVNI_GRAD(Santjago,Čile)
JE_GLAVNI_GRAD(Tokio,Japan)
JE_GLAVNI_GRAD(Beč,Austrija)
JE_GLAVNI_GRAD(Brazilija,Brazil)
JE_GLAVNI_GRAD(Karakas,Venecuela)
JE_GLAVNI_GRAD(Kairo,Egipat)
JE_GLAVNI_GRAD(Atina,Grčka)

Ova baza se može kreirati pomoću postupka opisanog u odeljku 5.2. na osnovu sledeće relacione baze podataka (sledećih relacija):

(CWA jednomesni)
JE_MEDITERAN_DRŽAVA({država}), ili
PODACI(predikat); *dom*(predikat)={"JE_MEDITERAN_DRŽAVA(Portugalija)"...}

(OW jednomesni)
IMA_IZLAZ_NA_MORE({država}), ili
PODACI(predikat); *dom*(predikat)={"IMA_IZLAZ_NA_MORE(Engleska)"...}

(CW dvomesni)
GL_GRAD_MEDIT_DRŽ({grad,država},{država}), ili
PODACI(predikat); *dom*(predikat)={"GL_GRAD_MEDIT_DRŽ(Lisabon,Portugalija)"...}

(OW dvomesni)
JE_GLAVNI_GRAD({grad,država},{država}), ili
PODACI(predikat); *dom*(predikat) ={"JE_GLAVNI_GRAD(London,Engleska)"...}

Analizom sadržaja predikatske baze projektant zaključuje da su svojstva: ***država je mediteranska država*** i ***grad je glavni grad date mediteranske države*** potpuno opisana, odnosno, baza je kompletna u odnosu na njih. Zato se odgovarajući predikati: JE_MEDITERAN_DRŽAVA i GL_GRAD_MEDIT_DRŽ deklarišu za CWA-predikate.

Ostali predikati: IMA_IZLAZ_NA_MORE i JE_GLAVNI_GRAD ne smeju se proglasiti za CWA-predikate, jer baza u odnosu na njih nije kompletna (za svojstvo: ***država ima izlaz na more*** nisu navedena imena svih država koja imaju izlaz na more, kao i za svojstvo ***grad je glavni grad date države*** nisu navedena imena glavnih gradova svih država sveta).

Na osnovu tako projektovane baze podataka i CWA-predikata, BASELOG-sistem će uvek davati korektne odgovore na upit učenika.

ZATVORENI SVET (CLOSED WORLD)

Na primer, na upit:

Da li je Francuska mediteranska država?

odnosno, u formi BASELOG-a:

~JE_MEDITERAN_DRŽAVA(Francuska)&

odgovor BASELOG-sistema je DA, što se i očekuje, jer se taj podatak nalazi u bazi podataka.

Međutim, na upit:

Da li je Irak mediteranska država?

odnosno, u formi BASELOG-a:

~JE_MEDITERAN_DRŽAVA(Irak)&

odgovor BASELOG-sistema je NE.

Na ovaj upit, pod pretpostavkom postojanja iste baze činjenica, odgovori sistema su:

DATALOG: NE
PROLOG: No

što se i poklapa sa radom BASELOG-sisema, jer on u ovom slučaju radi u režimu zatvorenog sveta, dakle na isti način kao i DATALOG i PROLOG.

OTVORENI SVET (OPEN WORLD)

Na upit:

Da li Nemačka ima izlaz na more?

odnosno u formi BASELOG-a:

~IMA_IZLAZ_NA_MORE(Nemačka)&

odgovor BASELOG-sistema je DA, što se i očekuje, jer se taj podatak nalazi u bazi.

Međutim, na upit:

Da li Finska ima izlaz na more?

odnosno, u formi BASELOG-a:

~IMA_IZLAZ_NA_MORE(Finska)&

odgovor BASELOG-sistema je NIJE IZVODIVO (ne znam), jer se taj podatak ne nalazi u bazi podataka, a BASELOG-sistem u ovom slučaju radi u režimu otvorenog sveta.

Na ovaj upit, pod pretpostavkom postojanja iste baze činjenica, odgovori sistema su:

DATALOG: NE
PROLOG: No (što se može tumačiti i kao neizvesno)

U ovom slučaju odgovor DATALOG-a je netačan i pedagoški neprihvatljiv, a odgovor PROLOG-a je neodređen, ali pedagoški prihvatljiv.

ZATVORENI SVET (CLOSED WORLD)

Na upit:

Da li je Ankara glavni grad mediteranske države Turske?

odnosno, u formi BASELOG-a:

~GL_GRAD_MEDIT_DRŽ(Ankara,Turska)&

odgovor BASELOG-sistema je DA, što se i očekuje, jer se taj podatak nalazi u bazi podataka.

Međutim, na upit:

Da li je Kazablanka glavni grad mediteranske države Maroko?

odnosno, u formi BASELOG-a:

~GL_GRAD_MEDIT_DRŽ(Kazablanka,Maroko)&

odgovor BASELOG-sistema je NE.
Na ovaj upit, pod pretpostavkom postojanja iste baze činjenica, odgovori sistema su:

DATALOG: NE
PROLOG: No (što se može tumačiti kao *neizvesno*).

što se i poklapa sa radom BASELOG-sisema, jer on u ovom slučaju radi u režimu zatvorenog sveta, dakle na isti način kao i DATALOG i PROLOG.

OTVORENI SVET (OPEN WORLD)

Na upit:

Da li je Tokio glavni grad Japana?

odnosno u formi BASELOG-a:

~JE_GLAVNI_GRAD(Tokio,Japan)&

odgovor BASELOG-sistema je DA, što se i očekuje, jer se taj podatak nalazi u bazi.

Međutim, na upit:

Da li Njujork glavni grad SAD?

odnosno u formi BASELOG-a:

~JE_GLAVNI_GRAD(Njujork,SAD)&

odgovor BASELOG-sistema je NIJE IZVODIVO (ne znam), jer se taj podatak ne nalazi u bazi podataka, a BASELOG-sistem u ovom slučaju radi u režimu otvorenog sveta.

Na ovaj upit, pod pretpostavkom iste baze činjenica, odgovori sistema su:

DATALOG: NE
PROLOG: No (što se može tumačiti i kao neizvesno)

U ovom slučaju odgovor DATALOG-a je netačan i pedagoški neprihvatljiv, a odgovor PROLOG-a je neodređen, ali pedagoški prihvatljiv.

Upiti, koji su do sada razmatrani, ilustruju neslaganje odgovora BASELOG-a i DATALOG-a usled različite primene CWA-principa (selektivno u BASELOG-u i neselektivno, odnosno, generalno u DATALOG-u). Međutim, zbog tretmana negacije kao konačnog neuspeha, i u PROLOG-u može doći do neslaganja odgovora BASELOG-a i PROLOG-a. To ilustruju sledeći upiti.

Neka je predikat *kontinentalna_država* određen sledećom aksiomom:

kontinent_drž(X1) :- not ima_izlaz_na_more(X1).

Na upit "Da li je Finska kontinentalna država?" PROLOG-sistem daje odgovor ***yes***, što je pogrešno! Ovaj odgovor PROLOG-sistem daje nakon što u generisanom cilju ***not ima_izlaz_na_more(finska)***, element ***ima_izlaz_na_more(finska)*** trpi konačan neuspeh (te činjenice nema u bazi podataka), pa se prihvata da je cilj ***not ima_izlaz_na_more(finska)*** zadovoljen.

DATALOG će zbog CWA-principa dati isti odgovor kao PROLOG, odnosno, ***yes***, što je pogrešno.

S obzirom da "*kontinentalna_država*" nije deklarisano za CWA-predikat, odgovor BASELOG-a je NIJE IZVODIVO (ne znam), što je pedagoški prihvatljivo.

Neslaganja odgovora BASELOG-a sa DATALOG-om i PROLOG-om javljaju se i u slučaju kada upit eksplicitno sadrži negaciju predikata.

Na upit "Da li Finska nema izlaz na more?" odgovor BASELOG-a je NIJE IZVODIVO, jer to nije CWA-predikat i tog podatka nema u bazi podataka.

Odgovor PROLOG-a na ovaj upit u formi: ?- *not ima_izlaz_na_more(finska)* je ***yes***, jer *ima_izlaz_na_more(finska)* trpi konačan neuspeh, pa se *not ima_izlaz_na_more(finska)* prihvata kao tačno.

Slično, DATALOG zbog CWA-principa generiše odgovor DA, što je pogrešno.

Na kraju, razmotriće se problem sinonima.

Na upit "Da li je Francuska primorska zemlja?" dobijaju se sledeći odgovori:

BASELOG: NIJE IZVODIVO (predikat *primorska_zemlja* je nepoznat sistemu)
DATALOG: NE (jer *primorska_zemlja(francuska)* nije izvodivo)
PROLOG: No (ne može da zadovolji cilj *primorska_zemlja(francuska)*)

Rešenje problema sinonima može se potražiti uvođenjem dodatne aksiome, koja povezuje predikate sistema sa sinonimom:

primorska_zemlja(X1) :- ima_izlaz_na_more(X1).

Sada, uz ovu aksiomu, dobijaju se sledeći odgovori:
BASELOG: DA
DATALOG: DA
PROLOG: Yes.

6. ZAKLJUČAK

U radu su opisane teoretske osnove za razvoj sistema, koji objedinjuje koncepte otvorenog i zatvorenog sveta u rukovanju bazama podataka u režimu otvorenog, zatvorenog i delimično otvorenog-zatvorenog sveta. Osnovni cilj je bio izgraditi sistem, koji neće raditi samo na osnovu jednog principa, nego će imati mogućnost prilagođavanja režima u toku rada.

Pored teoretskih koncepata, razvijen je i opisan konkretan programski sistem BASELOG. Ovaj sistem je proširenje postojećeg ADT sistema, koji je zasnovan na uređenoj linearnoj rezoluciji sa markiranim literalima, ali bez potrebe zadavanja aksioma, kojima bi se kompletirao prostor traženja rešenja.

Osnovna karakteristika programskog sistema BASELOG je ***fleksibilno rukovanje*** realno postojećim bazama podataka u željenom režimu rada. Ovo se odnosi i na klasične baze podataka i na baze znanja, i to naročito na baze znanja, koje se koriste u projektovanju obrazovnog računarskog softvera. Sistem za rukovanje bazama podataka, koji se koristi u projektovanju obrazovnog softvera, mora uvek da daje pedagoški prihvatljive odgovore na upit učenika. Navedeni su primeri na osnovu kojih se vidi da samo koncept zatvorenog sveta u bazama podataka u obrazovnom softveru nije prihvatljiv.

Fleksibilnost rada BASELOG sistema omogućena je uvođenjem liste CWA-predikata. Putem ove liste, korisnik (projektant nastavne baze podataka) definiše za koje će predikate sistem raditi u režimu zatvorenog sveta (klasične baze podataka), a za koje će predikate raditi u režimu otvorenog sveta (kao ADT sistem). U određivanju liste CWA-predikata projektant baze podataka BASELOG-sistema mora dobro da poznaje semantiku podataka, odnosno, kompletnost opisa predikata u bazi podataka, jer od toga zavisi kako će koji predikat biti deklarisan. U tom smislu, u radu je predložen princip za određivanje CWA-predikata.

Opisani su postupci putem kojih se vrši povezivanje BASELOG-sistema sa konkretnom bazom podataka (u pitanju je baza podataka kreirana putem FoxPro softvera za personalne računare). Omogućeno je povezivanje BASELOG-sistema sa relacionom bazom podataka u kojoj su domeni obeležja standardni tipovi podataka ili su domeni obeležja već formirani predikati.

Izvršena je komparacija rada BASELOG-sistema sa jezicima logičkog programiranja: PROLOG-om i DATALOG-om (kao paradigmom logičkog programiranja u oblasti baza podataka) i istaknute su prednosti BASELOG-sistema. U odnosu na DATALOG, prednost se ogleda u mogućnosti izbora režima rada: otvoren/zatvoren svet, a u odnosu na PROLOG, BASELOG-sistem je oslobođen nedostatka u odnosu na treman negacije i koncepciju konačnog neuspeha, jer se zasniva na logički korektnom dokazivaču teorema.

BASELOG-sistem se, takođe, može koristiti u nastavi i kao upitni jezik nad bazama podataka za dobijanje više različitih odgovora na upit i kao sistem za otkrivanje protivrečnosti u izjavama učenika.

Pravci daljeg razvoja BASELOG-sistema odnose se na razvoj posebnog korisničkog interfejsa za davanje semantičkih odgovora sa ciljem da se korisniku omogući korišćenje sistema bez potrebe za poznavanjem unutrašnjih mehanizama BASELOG-sistema. Takođe, moguće je realizovati ekstrahovanje samo željenih podataka iz baze (pretraživanje u bazi po zadatom / željenom kriterijumu, pre nego što se pozovu procedure za formiranje aksioma tipa BAKS). Moguće je dalje razraditi i formiranje aksioma tipa BAKS na osnovu podataka iz baze podataka, ali tako da te podatke odredi sam BASELOG-sistem, na osnovu datih kriterijuma.

Zahvaljujući osnovnom principu na kome je razvijen BASELOG-sistem: ***integracija koncepata otvorenog i zatvorenog sveta u jedinstven sistem za rukovanje bazama podataka***, ovakav sistem je moguće koristiti ne samo u projektovanju obrazovnog softvera, nego i u svim realnim situacijama sa semantikom, koja zahteva ovakav princip.

Opisanim rezultatima potvrđene su osnovna i pomoćne hipoteze, koje su formulisane u radu, i omogućeno je razvijanje konkretnih programskih aplikacija za potrebe obrazovanja i šire.

7. LITERATURA

[Al84] Alagić S, *Relacione baze podataka*, Svjetlost, Sarajevo, 1984.

[AT91] Alessi M. Stephen, Stanley R. Trollip, *Computer-based Instruction*, Methods and Development, 1991.

[Bra86] Bratko I, *Prolog Programming for Artifical Intelligence*, Addison-Wesley, 1986.

[Ber97] Berković I, *Deduktivne osnove za razvoj opisnih jezika logičkog programiranja*, Doktorska disertacija, Univerzitet u Novom Sadu, Tehnički fakultet "M. Pupin", Zrenjanin, 1997.

[Ber94] Berković I, *Varijabilne strategije pretraživanja u nastavno - orijentisanom sistemu za automatsko dokazivanje teorema*, Magistarski rad, Univerzitet u Novom Sadu, Tehnički fakultet "M. Pupin", Zrenjanin, 1994.

[BM91] Bertino E, Martino L, *Object-Oriented Database Management Systems: Concepts and Issues*, IEEE Computer, April 1991, pp. 33-47

[BA94] *Building Applications*, Microsoft ACCESS, Relational Database Management System for Windows, Microsoft Corporation, 1994.

[CGT89] Ceri S, Gottlob G, Tanza L, *What You Always Wanted to Know About Datalog (And Never Dared to Ask)*, IEEE Transactions on Knowledge and Data Engineering, Vol. 1, No. 1 March 1989, pp. 146-167

[Ch95] Chakravarthy S, *Early Active Database Efforts: A Capsule Summary*, IEEE Transactions on Knowledge and Data Engineering, Vol. 7, No. 6, December 1995, pp. 1008-1010

[Che76] Chen P. P, *The entity-relationship model: Towards a unified view of data*, ACM Transactions od Database Systems, Vol. 1, No. 1, March 1976, pp. 9-36

[Cu91] Cube von F, *Kibernetička i informacijskoteorijska didaktika*, Naša škola, Sarajevo, 1991, 7-8, str. 431-441

[Cv87] Cvetković D, *Diskretne matematičke strukture*, Naučna knjiga, Beograd, 1987.

[CER90] Czejdo B, Elmasri R, Rusinkiewicz M, *A Graphical Data Manipulation Language an Extended Entity-Relationship Model*, IEEE Computer, 1990, pp. 26-36

[ĆNR96] Ćurić N, Nadrljanski Đ, Radulović B, *Projektovanje automatizovanog testiranja znanja učenika*, VI Međunarodna konferencija "Informatika u obrazovanju i nove informacione tehnologije", Apatin 1996, str. 56-61

[Ču89] Čubrilo M, *Matematička logika za ekspertne sisteme*, Informator, Zagreb, 1989.

[Da87] Date C. J, *A Guide to the SQL Standard*, Addison-Wesley Publishing Company, 1987.

[Dev94] Devedžić V, *Objektno-orijentisane baze znanja*, Časopis INFO, Beograd 1994, 6/94, str. 12-19

[El84] Elsom - Cook T. M, *Design Considerations of an Intelligent Tutoring System for Programming Languages*, 1984.

[FT95] Fraternali P, Tanca L, *A Structured Approach for the Definition of the Semantics of Active Databases*, ACM Transactions on Database Systems, Vol. 20, No. 4, December 1995, pp. 414-471

[GE95] Gal A, Etzion O, *Maintaining Data-Driven Rules in Databases*, IEEE Computer, Vol. 28, No. 1, January 1995, pp. 28-38

[Gray84] Gray M. D. P, *Logic, Algebra and Databases*, Ellis Horwood Limited, England, 1984.

[Hen96] Hentzen W, *Visaul FoxPro 3.0*, Mikro knjiga, Beograd, 1996.

[HP88] Hotomski P, Pevac I, *Matematički i programski problemi veštačke inteligencije u oblasti automatskog dokazivanja teorema*, Naučna knjiga, Beograd, 1988.

[HK92] Hotomski P, Kujačić M, *Matematička logika i principi programiranja*, Tehnički fakultet "M. Pupin", Zrenjanin, 1992.

[Hot92] Hotomski P, *Deduktivnij podhod k avtomatičeskomu poroždeniu kombinatornih raspoloženii*, Proc. VI conf. LIRA '92, Novi Sad, 1992.

[Hot95] Hotomski P, *Sistemi veštačke inteligencije*, Tehnički fakultet "M. Pupin", Zrenjanin, 1995.

[Hot95a] Hotomski P, *Uloga sistema automatskog rezonovanja u dijalogu učenika i računara*, V Međunarodna konferencija "Informatika u obrazovanju i nove informacione tehnologije", Novi Sad 1995, str. 61-65

[Hot97] Hotomski P, *Deduktivni prilaz optimizaciji kombinatornih rasporeda pod restriktivnim uslovima*, XXIV Jugoslovenski simpozijum o operacionim istraživanjima SYMOPIS '97, Bečići, 1997.

[Il78] Ilić D. i dr, *Fizika za VIII razred osnovnog vaspitanja i obrazovanja*, Zavod za izdavanje udžbenika, Novi Sad, 1978.

[IYI96] Ishikawa H, Yamane Y, Izumida Y, Kawato N, *An Object-Oriented Database System Jasmine: Implementation, Application and Extension,* IEEE Transactions on Knowledge and Data Engineering, Vol. 8, No. 2, April 1996, pp. 285-303

[JD96] Jerinić Lj, Devedžić V, *Realizacija studentovog modela u EduSof-u*, VI Međunarodna konferencija "Informatika u obrazovanju i nove informacione tehnologije", Apatin 1996, str. 90-95

[Ku97] Kutlača Đ, *Informacione tehnologije u obrazovnom sistemu: izazov i potreba, rizik i investicija u budućnost*, INFO Science, Beograd, 3/1997, str. 18-23

[Kv78] Kvaščev R, *Modeliranje procesa učenja,* Prosveta, Beograd, 1978.

[La75] Landa L. N, *Teorijski problemi algoritmizacije i heuristike u nastavi*, Časopis Pedagogija, Beograd, 4/1975, str. 377-391

[LNB93] Lazarević B, Nešković S, Bataveljić P, *Objektno-orijentisana transformaciona metoda razvoja informacionih sistema*, Publikacija Laboratorije za informacione sisteme FON-a, Beograd, 1993.

[La96] Lazarević B, i dr, *Formiranje i pretraživanje baza podataka u sistemu naučnih i tehnoloških informacija Srbije*, Ministarstvo za nauku i tehnologiju Republike Srbije, Beograd, 1996

[LM93] Lazarević D. S, Marjanović Z, *Upravljanje znanjem u POSTGRES-u*, XX Jugoslovenski simpozijum o operacionim istraživanjima SYMOPIS '93, Beograd 1993, str. 91-94

[LaS95] Lazarević D. S, *Objektno orijentisane karakteristike postrelacionog sistema Postgres*, XXII Jugoslovenski simpozijum o operacionim istraživanjima SYMOPIS '95, Gornji Milanovac 1995, str. 195-198

[Le76] Lekić Đ, *Eksperimentalna didaktika,* Pedagoško-tehnički fakultet, Zrenjanin, 1976.

[LL95] Levene M, Loizon G, *A Graph-Based Data Model and its Ramifications*, IEEE Transactins on Knowledge and Data Engineering, Vol. 7, No. 5, October 1995, pp. 809-823

[LM93] Lujić S, Marjanović Z, *Sistem pravila - aktivna Starburst komponenta*, XX Jugoslovenski simpozijum o operacionim istraživanjima SYMOPIS '93, Beograd 1993, str. 95-98

[LuW90] Luk W. S, *Building Natural Language Interface to an ER database*, Entity-Relationship Approach to Database Design and Querying, Elsevier Science Publishers, 1990, pp. 345-360

[Lu96] Luković I, *Integracija šema modula baze podataka informacionog sistema*, Doktorska disertacija, Univerzitet u Novom Sadu, Fakultet tehničkih nauka, 1996.

[LHRB95] Luković I, Hotomski P, Radulović B, Berković I, *Dokaz zadovoljenja generalizovanih zavisnosti podataka metodom automatskog rezonovanja*, II Simpozijum o računarskim naukama i informatici, Brezovica, YUINFO '95, 1995.

[LHRB97] Luković I, Hotomski P, Radulović B, Berković I, *A Technique for the Implicational Problem Resolving for Generalized Data dependencies*, Proceedings of VIII International Conference on Logic and Computer Science LIRA '97, Novi Sad 1997, pp. 111-119.

[MMT89] Malhotra A, Markowitz M. H, Tsalalikhin Y, Pazel D, Burns L, *An Entity-Relationship Programming Language*, IEEE Transactions on Software Engineering, Vol. 15, No. 9, September 1989, pp. 1120-1130

[MK96] Mandić D, Krsmanović S, *Generisanje multimedijalnog obrazovnog softvera*, VI Međunarodna konferencija "Informatika u obrazovanju i nove informacione tehnologije", Apatin, 1996, str. 109-112

[Ma90] Marjanović Z, *ORACLE relacioni sistem za upravljanje bazom podataka*, Beograd, 1990.

[Ma93] Marjanović Z, *Aktivni sistemi baza podataka*, XX Jugoslovenski simpozijum o operacionim istraživanjima SYMOPIS '93, Beograd 1993., str. 103-106

[Ma96] Marjanović Z, *Logička specifikacija i logička prototipska implementacija baze podataka i aplikacija*, Doktorska disertacija, Univerzitet u Beogradu, Fakultet organizacionih nauka, 1996.

[Ma97] Marjanović Z, *Prikaz karakteristika UNIFIED Method-a*, INFO Science, Beograd 1997, 3/97, str. 14-22

[Mi92] Milojković V, *Oracle (ver. 5) arhitektura i administracija*, Institut za nuklearne nauke u Vinči, Beograd, 1992.

[Mil96] Milošević M, *Geografija za 7. razred osnovne škole*, Zavod za udžbenike i nastavna sredstva, Beograd, 1996.

[Mo91] Mogin P, *Organizacija datoteka i uvod u baze podataka*, Viša škola za organizaciju i informatiku, Novi Sad, 1991.

[ML96] Mogin P, Luković I, *Principi baza podataka*, Fakultet tehničkih nauka i MP STYLOS, Novi Sad, 1996.

[Nad86] Nadrljanski Đ, *Kompjuteri, nastava i učenje,* MISAO, Novi Sad, 1986.

[Nad94] Nadrljanski Đ, *Obrazovni računarski softver*, Tehnički fakultet "M. Pupin", Zrenjanin, 1994.

[Nad96] Nadrljanski Đ, *Informacioni sistemi u obrazovanju*, Centar za usavršavanje rukovodilaca u obrazovanju, Beograd, 1996.

[Nad97a] Nadrljanski Đ, *Multimedije i virtuelna realnost u obrazovanju*, Tehnički fakultet "M. Pupin", Zrenjanin, 1997.

[Nad97b] Nadrljanski Đ, *Multimedije i virtuelna realnost u obrazovanju*, Časopis INFO, Beograd 1997, 2/97

[Nad97c] Nadrljanski Đ, *Virtuelna realnost kao osnova za projektovanje obrazovnog računarskog softvera*, Časopis Pedagogija, Beograd, 1997.

[Ni96] Nikolić M, *Za korak ispred*, BIGZ, Časopis Računari, 1996, br. 112, str. 89-93

[OB91] O'Brien K. S, *Turbo Pascal 6.0, kompletan vodič*, Mikro knjiga, Beograd, 1991.

[OL71] *Opšta enciklopedija Larousse*, Izdavačka kuća "Vuk Karadžić", Beograd, 1971.

[OS95] Ozsoyoglu G, Snodgrass R. T, *Temporal and Real - Time Databases: A Survey*, IEEE Transactions on Knowledge and Data Engineering, Vol. 7, No. 4, August 1995, pp. 513-532

[PBG89] Paredaens J, Bra D. P, Gyssens M, Gucht V. D, *The Structure of the Relational Database Model*, Springer-Verlag, Berlin Heidelberg, 1989.

[PS85] Parent C, Spaccapierta S, *An Algebra for a General Entity-Relationship Model*, IEEE Transactions on Software Engineering, Vol. Se-11, No. 7. July 1985, pp. 634-642

[Pe96] Perović M, *Istorija za 7. razred osnovne škole*, Zavod za udžbenike i nastavna sredstva, Beograd, 1996.

[Pol91] Poliščuk J, *Softver četvrte generacije ORACLE*, Tehnička knjiga, Beograd, 1991.

[Po76] Popov E. V, Firdman G. R, *Algoritmičeskie osnovi intellektualnih robotov i iskusstvennogo intellekta*, Nauka, Moskva, 1976.

[PBR90] Premeriani W. J, Blaha M. R, Rumbaugh J. E, Varwig T. A, *An Object-oriented Relational Database*, Communications of the ACM, Nov. 1990.

[Pro94] Prohaska D, *Deduktivna metoda generisanja kombinatornih rasporeda i njena programska algoritmizacija*, Magistarski rad, Univerzitet u Novom Sadu, Tehnički fakulet "M. Pupin", Zrenjanin 1994.

[Rač97] RAČUNARI, BIGZ, Beograd 1997, (4/97, 5/97)

[Ra87] Radovan M, *Programiranje u Prologu*, Zagreb, 1987.

[RB95] Radulović B, Berković I, *Relacione i objektne baze podataka*, V Međunarodna konferencija "Informatika u obrazovanju i nove informacione tehnologije", Novi Sad 1995, str. 77-81

[Rad96] Radulović B, *Pregled novih metoda u postupku logičkog projektovanja baza podataka*, VI Međunarodna konferencija "Informatika u obrazovanju i nove informacione tehnologije", Apatin 1996, str. 183-187

[Ro65] Robinson J. A, *A machine oriented logic based on resolution principle*, J. ACM, Vol. 12, 1965, pp. 23-41

[Si95] Simpson A, *Access 2*, Mikro knjiga, Beograd, 1995.

[SRH90] Stonebraker M, Rowe L. A, Hirohama M, *The Implementation of POSTGRES*, IEEE Transactions on Knowledge and Data Engineering, Vol. 2, No. 1, March 1990, pp. 125-142

[Sto90] Stonebraker M, *Future Trends in Database Systems*, IEEE Transactions on Knowledge and Data Engineering, Vol. 2, No. 1, March 1990, pp. 33-44

[Sto92] Stonebraker M, *The integration of rule systems and database systems*, IEEE Transactions on Knowledge and Data Engineering, Vol. 4, No. 4, October 1992, pp. 415-423

[SNV95] Subrahmanian S.V, Nau D, Vago C, *WFS + Branch and Bound = Stable Models*, IEEE Transactions on Knowledge and Data Engineering, Vol. 7, No. 3, June 1995, pp. 362-377

[Te89] Tejlor, *Information Technologies and Basic Learning: Reading, Writing, Science and Mathematics*, Paris, 1989.

[Ullm85] Ullman J. D, *Implementation of Logical Query Languages for Databases*, ACM Transactions on Database Systems, Vol. 10, No. 3, September 1985, pp. 289-321

[Ullm88] Ullman J. D, *Principles of Databases and Knowledge-Base Systems*, Vol. I, Computer Science Press, New York, 1988.

[Va96] Vadaparty K, *Developing an ODBMS application: Basic steps*, ODBMS, January 1996, pp. 19-26

[Vos95] Voskresenski K., *Modeli učenja i nastave u didaktičkim softverima*, V Međunarodna konferencija "Informatika u obrazovanju i nove informacione tehnologije", Novi Sad 1995, str. 143-145

[Vra94] Vraneš S., *BEST kao alat za izgradnju sistema za primene u obrazovanju*, IV Konferencija "Informatika u obrazovanju i nove informacione tehnologije", Novi Sad 1994, str. 131-139

[WLH90] Wilkinson K, Lyngback P, Hason W, *The IRIS Architecture and Implementation*, IEEE Transactions on Knowledge and Data Engineering, Vol. 2, No. 1, March 1990, pp. 63-75

9. INDEKS