ER JEZIK: PRIJEDLOG PROŠIRENJA

INFORMATICA 3/91

Keywords: entity-relatinship data model, data modeling, logical database design

Mario Radovan
Fakultet Ekonomije i turizma, Pula;
Institut »Jožef Stefan«, Ljubljana

ER LANGUAGE: A PROPOSAL FOR EXTENSION The paper gives an analysis of the expressive power and of (some) limitatins of ER language as a means for designing data models. Starting from a different "origin", "nature" and function of different entities - and from the practical need that such a features be emphasized in the model - two proposals for extending the "standard" ER language are given: symbols for *process* and *disjunction* are introduced.

ER JEZIK: PRIJEDLOG PROŠIRENJA U članku je data analiza izražajnih mogućnosti i (nekih) ograničenja" ER jezika kao sredstva za oblikovanje modela podataka. Polazeći od različitog "porijekla", "prirode" i funkcije pojedinih entiteta - i praktikče potrebe da se ta svojstva naglase u modelu - iznijeta su dva prijedloga proširenja "standardnog" ER jezika: uvedeni su simboli *procesa* i *disjunkcije*.

1. Uvod

Model (baze) podataka tvori esencijalni dio informacijskog sistema, i *presudno* utječe na njegov kvalitet. Stoga je tokom posljednjih decenija razvijen niz prijedloga jezika (pretežno grafičkih) za oblikovanje modela (baze) podataka. Dominantno mjesto i najširu primjenu među njima ima *ER jezik* (ili: *ER model podataka*, kako se češće naziva u literaturi), čija je osnova predložena u <Che 76>.

"Nešto lakonskiji" pristup oblikovanju baze podataka možemo naći, naprimjer, u <Per 89>. Tamo čitamo: "Relacijska teorija zasnovana je na zdravom razumu. ... Ako 'zdravo razmišljate' (use sound thinking) kod oblikovanja svojih tabela, onda nećete imati problema kod kasnijeg 'prikazivanja' svojih podataka". Autorima ove "teorije oblikovanja baze podataka" oprostiti ćemo na "nepodnošljivoj lakoći" (i kratkoći) njihove teorije, jer je knjiga "Understanding Oracle" ipak prije svega priručnik za sistem Oracle. (No, takva "teorija" zacijelo može imati (a po svemu sudeći i ima!) nepovoljan utjecaj na modele podataka koje oblikuju njeni čitatelji!)

U cilju izbjegavanja mogućih (a nepotrebnih) "terminoloških prijepora", pogledajmo najprije (uobičajeno) značenje samog pojma "model podataka".

Prema Ullmanu, <Ull 88>, modelom podataka nazivamo "matematički formalizam" sačinjen od dva dijela:

1. Notacije (i metode) za opis *forme* podataka.
2. Skupa operacija za rukovanje *podacima*.

Obzirom da u ER jeziku nisu definirane "operacije" (niti se taj jezik eksplicitno bavi samim podacima, već samo njihovim formama) Ullman zaključuje da možemo tvrditi kako "ER model" zapravo i nije "model podataka".

Formalno, Ullmanova primjedba stoji, jer ER jezik ne ispunjava uvjet 2. Stoga ovdje i preferiramo izraz "ER jezik" (kojim se opisuju *forme* entiteta iz "fragmenta realnog svijeta" o kojem želimo "bilježiti neke podatke"). Utoliko bi i zapise u ER jeziku valjalo zvati "modelima *forme* podataka" (što ER model, zapravo, i jeste). No, radi jednostavnosti izražavanja, u nastavku koristimo izraz "ER model podataka" (kako je to, uostalom, u literaturi i uobičajeno).

Analiza i prijedlozi koje iznosimo u nastavku, uvelike temelje na iskustvima primjene ER jezika u okviru projektiranja baza podataka za potrebe turističke privrede (te su, stoga, iz tog problemskog područja uzimani i primjeri).

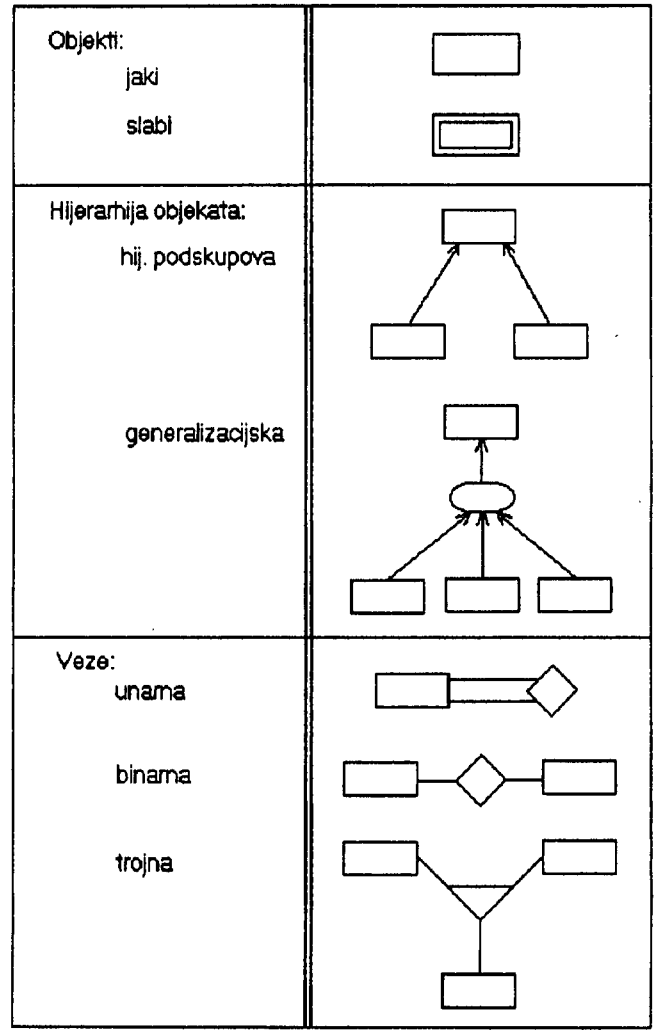
Polazeći od tih iskustava - te od "standardnog" ER jezika (datog u npr. <Teo 86>, <Dat 90> ili <Rad 91>) - ovdje predlažemo ono što držimo *stvarnom potrebom* a ne (samo) hipotetičkom *mogućnošću* proširenja ER jezika.

2. O ER jeziku

Polaznu osnovu ER jezika možemo izreći otprilike slijedećim riječima: Podaci su znanja o *objektima*, *vezama* (među tim objektima) i *svojstvima* (objekata odnosno veza); stoga, cilj jezika za predstavljanje strukture podataka jeste da omogući precizan i jednostavan zapis znanja tih triju temeljnih kategorija. (Napomenimo, da pojmom "entitet" ovdje ne označavamo samo "objekte" - kako se to često čini - već je to "generativni pojam" koji obuhvaća sve tri iznad navedene kategorije.)

Na Chenov prijedlog uslijedile su brojne dopune (proširenja) jezika, posebno uvođenjem n-arnih veza za n > 2, i hijerarhije entiteta. Obzirom na terminološki i notacijski nesklad pojedinih prikaza jezika, na slici 1 data je osnovna notacija ER jezika kakvu koristimo u ovom tekstu.

Pritom, imena objekata i veza upisujemo u sam simbol, a njihova svojstva zapisujemo pored simbola (kako to ilustriraju slijedeće slike). Isto tako, u simbol upisujemo i "svojstvo generalizacije" (kod generalizacijske hijerarhije). Identifikatore objekata podcrtavamo.



Slika 1

Prema Thompsonu, <Tho, 89>, ER jezik je "izuzetno koristan za *razumijevanje i oblikovanje* baze podataka ... Njegova temeljna odlika

jeste u tome što "prekida" sa razmišljanjem na razini "zapisa" (records)", i okreće se ka *entitetima* sistema. Naime, " ... zapisi su sasma beskorisni (na razini oblikovanja baze), dok su entiteti i veze (među njima) "obećavajući početak". Stoga je od suštinskog značaja "zahvatiti" upravo njih". U svom vrlo elokventnom stilu (više efektnom nego efikasnom), Thompson zaključuje: "Model podataka živi ili umire 'od lakoće' kojom se dade shvatiti". I koristiti! - dodali bismo. A dosadašnjih 15 godina široke primjene ER jezi- ka pokazuje da ER jezik - "živi". Čini se da bismo iz toga onda smjeli izvesti i zaključak o "lakoći njegova shvaćanja i primjene".

ER jeziku pripisuju se odlike i kao komunikacijskom jeziku. "ER jezik pokazao se najuspješnijim kao *sredstvo komuniciranja* između projektanta i korisnika u toku analize (sistema) i oblikovanja baze podataka", nalazimo u <Teo 86>. Ta odlika, prema Teorey et al, proizlazi iz njegove "okrenutosti entitetima" (koju ističe i Thompson), te jednostavnosti samog jezika.

Ostajući uvelike suglasni sa iznijetim pozitivnim ocjenama ER jezika, u ovom članku želimo istaći njegove odlike kao sredstva za *zorno dokumentiranje* strukture baze podataka, ali i neka ograničenja koja "dolaze" iz nemogućnosti prikaza različitosti "porijekla" i "procesnih uloga" pojedinih entiteta modela.

Iznijeti prijedlozi proširenja standardnog ER jezika imaju za cilj otklanjanje tih ograničenja (nedostataka).

3. Izvorni i izvedeni entiteti

Prvi "raskorak" između ER jezika "na razini načela" i "na razini primjene" dolazi odtuda što, u praksi, među entitetima sistema postoji znatna razlika u prirodi nastanka (formiranja) i ulozi koju imaju u modelu (baze) podataka (odno- sno u procesu obrade podataka). S tog aspekta promatranja, držimo primjerenim (i potrebnim) entitete (tj., u samoj bazi: "tabele" i "kolone") podijeliti na slijedeći način:

Svojstva: *izvorna* i *izvedena*
Objekti i veze: *izvorni*, *izvedeni* i *kombinirani*.

Izvedenim (izračunatim) svojstvima nazvali smo ona svojstva čije se vrijednosti izračunavaju (izvode) iz vrijednosti drugih svojstava (istog ili pak nekih drugih objekata).

Izvornim svojstvima nazvali smo ona svojstva objekata koja nisu izvedena.

Izvedenim objektima (vezama) nazvali smo one objekte (veze) čija su sva svojstva izvedena.

Izvornim objektima (vezama) nazvali smo one objekte (veze) kod kojih nijedno svojstvo nije izvedeno. Takve objekte često nazivamo i matičnima.

Kombiniranim objektima (vezama) nazvali smo one objekte (veze) koji sadrže bar jedno (ali ne sve!) izvedeno svojstvo.

3.1. Izvedena svojstva

Jednostavan primjer izvedenog svojstva imali bismo kod (slabog) objekta STAVKA, sa svojstvima: ... KOLIČINA, CIJENA, IZNOS.

Očito, svojstvo IZNOS izvedeno je svojstvo jer se njegova vrijednost izračunava iz vrijednosti svojstava CIJENA i KOLIČINA. A to bi onda ujedno značilo da na relaciji dobivenoj "prijevodom" tako definiranog objekta STAVKA, postoji (i) funkcijska zavisnost

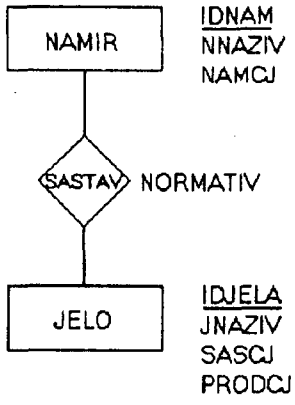
CIJENA,KOLIČINA --> IZNOS

Teorijski gledano, entiteti ne bi trebali (a ni smjeli!) sadržavati takova (izračunata) svojstva. Naime, takva svojstva, zapravo (u buduću tabelu) uvode redundantna polja, a time - u pravilu - i funkcijske zavisnosti koje nisu od ključa.

Prihvatimo da je u promatranom primjeru (i praktički gledano) odista nepotrebno eksplicitno "čuvati" vrijednosti svojstva IZNOS jer je CPU vrijeme za njegovo izračunavanje zanemarivo

malo u odnosu na vrijeme dostupa do samog zapisa u kojem su sadržane vrijednosti za njegovo izračunavanje. Drugim riječima, dostup do "iznosa" neznatno varira u zavisnosti da li ga samo čitamo ili ga i izračunavamo.

Međutim, sa aspekta ER modela, zanimljivijim izgleda problem prikaza izvedenosti onih svojstava čije se vrijednosti izračunavaju na temelju vrijednosti svojstava *drugih* objekata. Takav primjer dat je na slici 2.



Slika 2

Relacijski zapis tog modela glasi:

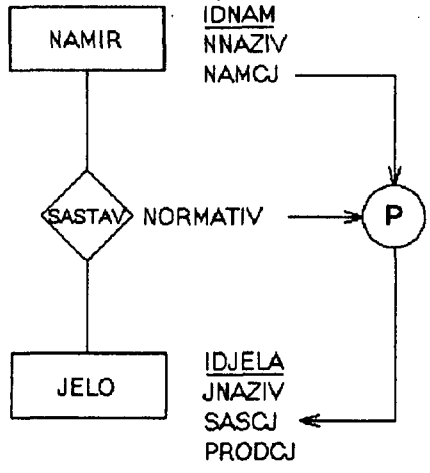
NAMIR(IDNAM,NNAZIV,NAMCJ)
SASTAV(IDNAM,IDJELA,NORMATIV)
JELO(IDJELA,JNAZIV,SASCJ,PRODCJ).

U ovom modelu, izvedeno svojstvo jeste SASCJ (cijena sastojaka (namirnica) u datom jelu). Vrijednost tog svojstva izračunava se iz vrijednosti (izvornog) svojstva NAMCJ (cijene namirnica) i izvornog svojstva NORMATIV veze SASTAV. Naime, normativ pokazuje količinu pojedine namirnice u pojedinom jelu.

Napomenimo da svojstvo PRODCJ (prodajna cijena jela) jeste izvorno svojstvo, jer se njegova vrijednost ne izračunava (algoritamski) iz cijene sastojaka (SASCJ). Naime, iako se vrijednost SASCJ svakako uzima kao osnova, ipak se vrijednost svojstva PRODCJ utvrđuje aproksimativno, i to u zavisnosti od niza faktora, poput energije i rada potrebnih za pripremu (i serviranje) jela, pouzdanosti ponude namirnica, potražnje jela, itd.

Vratimo se izvedenom svojstvu SASCJ. U razmatranom slučaju, pokazalo se potrebnim ("procesno opravdanim") uvođenje tog (redundantnog) svojstva u model podataka. Naime, to se svojstvo *često koristi*, i to ne samo za aproksimativno utvrđivanje prodajne cijene jela, već i kod raznih drugih kalkulacija. Bilo bi stoga neopravdano - zbog samog načelnog zahtjeva po neredundantnosti modela - to svojstvo izostaviti iz njega.

Međutim, ostavljajući po strani probleme redundance na razini relacijskog modela (tj. na razini "tabela podataka"), ovdje se postavlja pitanje da li i kako - u samom ER modelu - *eksplicitno prikazati* izvedenost svojstva i/ili način izračunavanja njegove vrijednosti. Drugim riječima, valja odlučiti o eventualnoj ekstenziji ER jezika uvođenjem (simbola) *procesa*. Na slici 3 dat je jedan prijedlog načina na koji se to može učiniti.



Slika 3

Čini se da bi takva ekstenzija jezika učinila model informativnijim (a da pritom ipak ne zađemo u DeMarcov dijagram toka podataka (DTP, <Rad 91>)). Međutim, kod opsežnijih modela, eksplicitnim prikazom procesne prirode svojstava (pose- bno ako izvedenih svojstava ima mnogo!) model bi mogao brzo postati nepreglednim. A upravo smo jasnoću i preglednost modela podataka isticali kao temeljnu odliku ER jezika

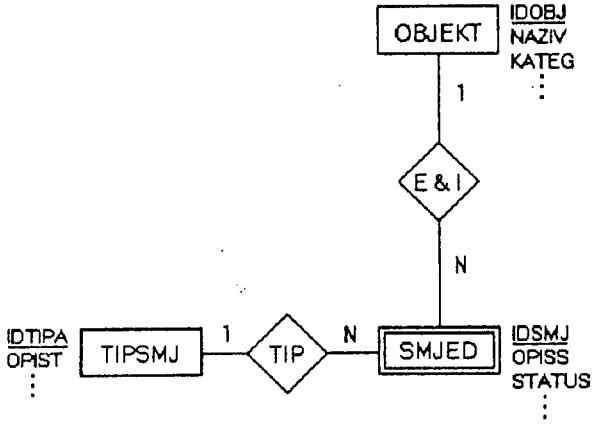
u ulozi sredstva za dokumentiranje. Stoga, prijedlog sa slike 3 - na razini izvedenih svojstava - navodimo samo kao *mogućnost*, ali ne i kao obavezan element ER jezika.

U nastavku, razmatramo potrebu (opravdanost) uvođenja simbola procesa na razini prikaza izvedenih objekata (veza).

3.2. Izvedeni objekti (veze)

Prema danim definicijama, objekt NAMIRnice i veza SASTAV jesu izvorni. Objekt JELO je kombiniran jer sadrži jedno izvedeno svojstvo (ali i dva izvorna svojstva).

No, čini se da posebnu pažnju zaslužuju izvedeni objekti (veze). Potrebu po uvođenju takvih entiteta analiziramo na primjeru situacije koju opisuje model podataka sa slike 4.

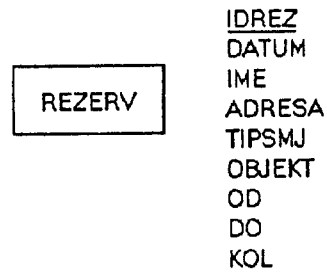


Slika 4

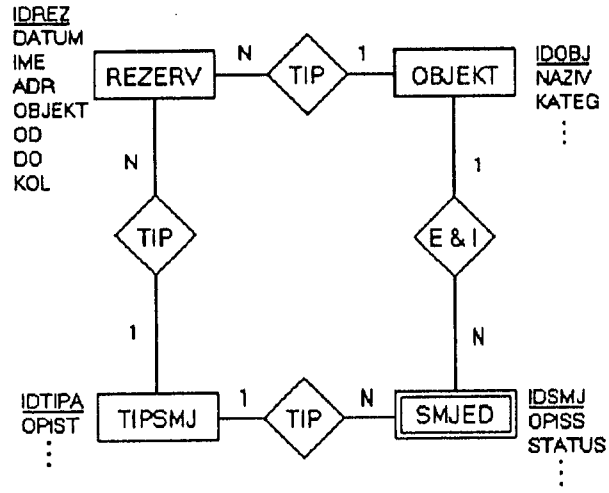
Model sadrži objekte: OBJEKT (hotel, turističko naselje, ...) SMJED (smještajna jedinica u okviru objekta) i TIPSMJ (tip smještajne jedinice, određen prema kapacitetu, komforu, ... pojedine smještajne jedinice).

Postavlja se pitanje da li takav model omogućava da se izvrši rezervaciju npr. 5 smještajnih jedinica datoga tipa. Uzmimo pritom da bi rezervacija mogla (morala) sadržavati barem svojstva data na slici 5.

Model sa slike 6 pokazuje na koji način je to *moguće* učiniti.



Slika 5

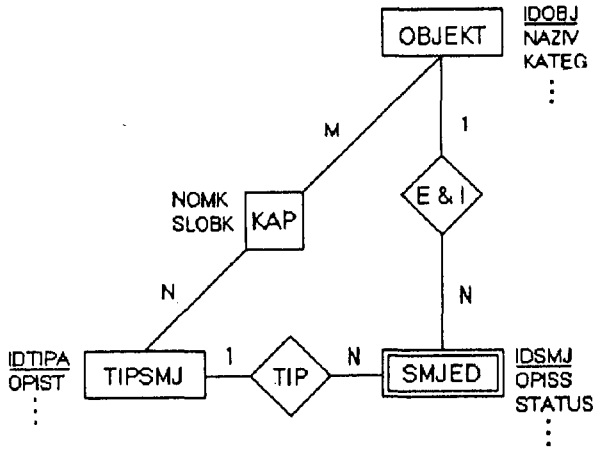


Slika 6

Međutim, već pokušaj provjere ispunjenosti prvog preduvjeta za *prihvaćanje* rezervacije - naime, postojanja slobodnih smještajnih jedinica! - ukazuje na izrazite slabosti takova rješenja. Naime, prema modelu podataka sa slike 6, uvid u "slobodne kapacitete" bio bi procesno vrlo zahtjevan. Jer postojanje slobodnih smještajnih jedinica (u datom objektu, datog tipa, u datom periodu) bilo bi moguće utvrditi jedino tako da se za dati objekt najprije izračuna broj smještajnih jedinica (datog tipa); da se zatim - iz tabele REZERVacije - utvrdi rezerviranost (zauzetost) jedinica toga tipa (za dati period), te da se "iz razlike" utvrdi postojanje i broj slobodnih smještajnih jedinica - i tek nakon toga izvrši rezervaciju (ako slobodni kapacitet to dopušta). Naravno, iako je takav postupka - u kontekstu datog

modela podataka - *moguć*, operativno ("procesno") promatrano, takav bi postupak izračunavanja *bio krajnje suboptimalan.*

Proces provjere prihvatljivosti (a i izvršenja) rezervacije dade se *bitno* pojednostaviti uvođenjem redundantnog entiteta (veze) KAPacitet. (Dakle, odstupanjem od jednog od temeljnih načela oblikovanja modela podataka, koje u <Teo 86> glasi: *"Redundantne veze trebaju biti eliminirane!"*) Pritom, kapacitetom nazivamo količinu smještajnih jedinica po tipovima i objektima. Dakle, KAPacitet može biti prikazan vezom (mnogo-mnogo), kako je to učinjeno na slici 7.



Slika 7

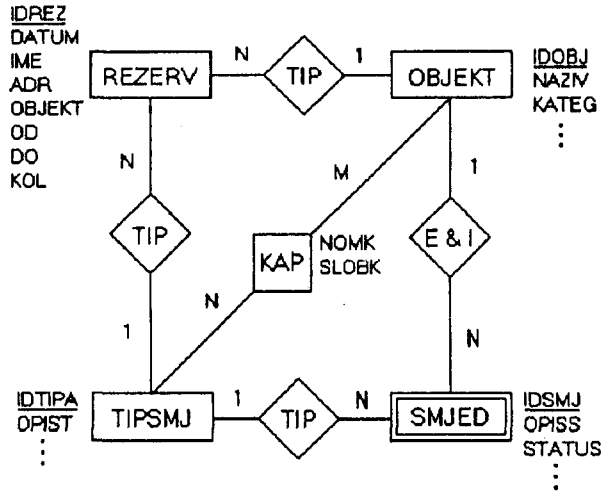
Napomenimo da je svojstvo NOMK (nominalni kapacitet) skalar, a da je svojstvo SLOBK (slobodni kapaciteti) vektor (sa 366 polja), pri *čemu svako sadrži broj slobodnih smještajnih* jedinica (datog tipa u datom objektu) za pripadni dan u godini. Rezervacija je sada, naravno, moguća ako - u traženom periodu - broj slobodnih smještajnih jedinica nije manji od broja traženih. A uz postojanje izvedene (redundantne) veze KAPaciteti, to je trivijalno utvrditi.

Naravno, prihvaćanje rezervacije uvjetuje i on-line ažuriranje vrijednosti svojstva SLOBK za dati period. (Napomenimo, da se i na razini smještajne jedinice vodi analogna evidencija (svojstvo STATUS), i to zbog omogućavanja rezerviranja točno određene smještajne jedinice. Nadalje, status zauzetosti sobe odnosno kapaciteta ne mijenja se samo na temelju rezervacije, *ali u te pojedinosti ovdje nema potrebe* zalaziti.)

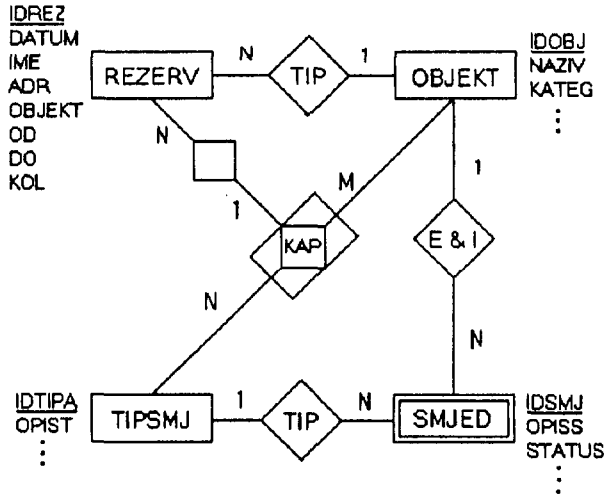
Na razini "zornog prikaza i dokumentacije", model podataka sa slike 7 pokazuje dvije slabosti.

1. Objekt REZERVacija ne možemo "vezati" na *vezu* KAPacitet. (Naime, veza može "stajati" samo između objekata.) Stoga smo prinuđeni postupiti na jedan od dva načina data na slici 8.

Za vezu rezervacije i kapaciteta sa slike 8a, možemo reći da je (u najboljem slučaju)



(a)



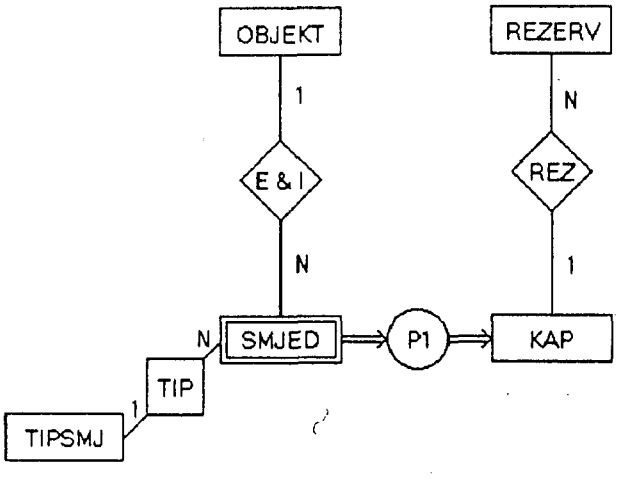
(b)

Slika 8

"implicitna". Naime, postojanje takve veze nipošto nije "očito" iz samog modela, već to postaje tek ako se "prisjetimo" da će se - kao vanjski ključevi - u shemi relacije REZERVacija naći svojstva IDOBJ i IDTIP, te da upravo taj par svojstava tvori identifikator (ključ) veze (relacije) KAPaciteti.

2. Uvođenje simbola *objekt-veza* (<Rad 91>), kako je to učinjeno na slici 8b, omogućuje eksplicitni pokaz vezanosti rezervacije na kapacitet, te stoga uvođenje takvog simbola držimo korisnim.

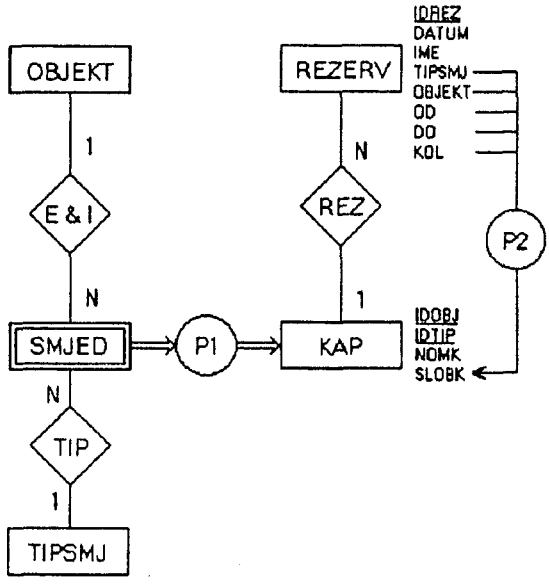
Međutim, nijedan od dvaju navedenih načina ne pokazuje *eksplicitno* da je entitet KAPaciteti *izvedeni* entitet! (Dakle, "formalno redundantan", ali "procesno potreban"!) Stoga, ovdje predlažemo uvođenje "procesnog simbola", koji je već "ispitivan" kod izvedenih svojstava, a čije uvođenje držimo sasma opravdanim kod izvedenih objekata odnosno veza. Naime, izgleda da se tim proširenjem ER jezika ne gubi ništa; model neće postati nepreglednijim, već upravo suprotno, kako to ilustrira slika 9.



Slika 9

Za izvedeni entitet KAPacitet možemo reći da - posredstvom procesa P - slijedi iz (slabog) objekta SMJED. Naime nominalni kapacitet (NOMK) - po smještajnim objektima i tipovima smještajnih jedinica - izračunava se na temelju svojstava objekta (budućih "kolona tabele") SMJED, što modeli sa slike 8 nisu eksplicitno pokazivali.

Razmotrimo i izvedeno svojstvo SLOBK (slobodni kapaciteti). Naime, učinivši "prvi korak" ka "procesualizaciji" modela podataka, postavlja se pitanje zašto ne učiniti i više. Naprimjer, zašto ne pokazati eksplicitno (u modelu) i "način utjecaja" rezervacije na slobodan kapacitet (SLOBK), kako je to učinjeno na slici 10?



Slika 10

Očito, ovakav način prikazivanja "porijekla i procesne prirode" izvedenih svojstava čini model podataka informativnijim. Međutim, u slučajevima gdje ima velik broj izvedenih svojstava, eksplicitan prikaz procesa (načina) njihova formira- nja, doveo bi do toga da model podataka postane nepreglednim.

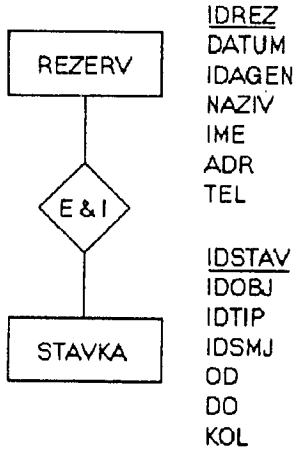
Stoga, uvođenje "procesnih opisa" na razini izvedenih *objekata/veza* držimo sasma opravdanim i korisnim. S druge strane, primjerenost uvođenja takova prikaza na razini *svojstava* zavisi od specifičnih osobina konkretnog modela podataka.

Konačno, želimo li imati oboje: informativnost i preglednost, model možemo prikazati na dvije razine detaljizacije. Dakle, na "global-

noj razini" (bez procesnih opisa za svojstva), i na "razini detalja" (sa takvim opisima).

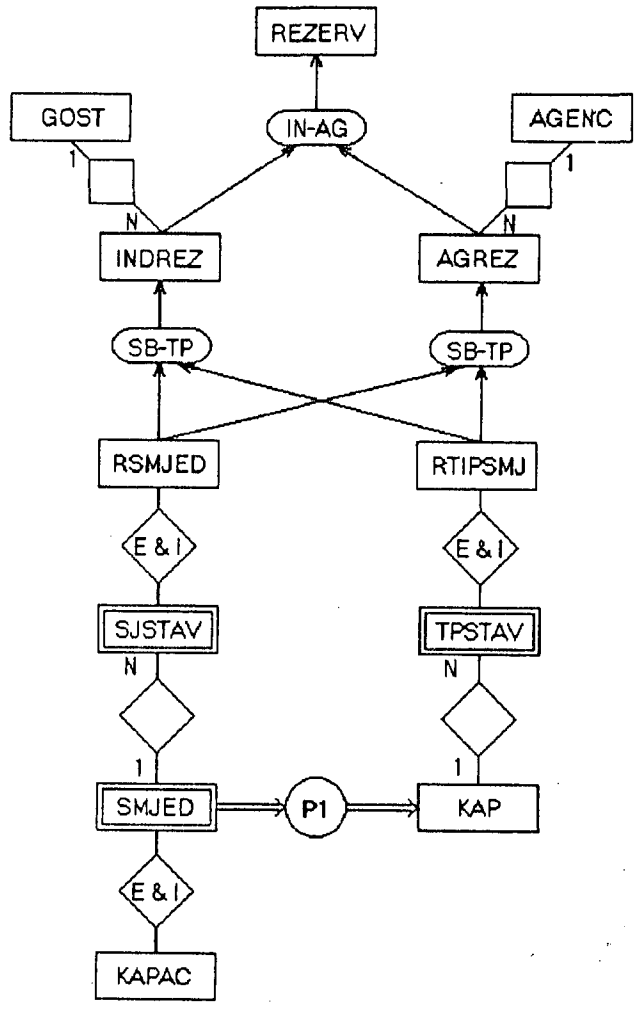
4. Veze i disjunkcija

Sistem rezervacija mora omogućavati rezervaciju ne samo datog tipa smještajne jedinice, već i točno određene smještajne jedinice. Modeli sa slika 6 - 10, očito to ne omogućavaju. Nadalje, rezerviranje može vršiti pojedinac ili agencija; isto tako, valja omogućiti da se jednom rezervacijom rezerviraju bilo razne smještajne jedinice (ili jedinice raznih tipova), bilo pojedinačnih smještajnih jedinica, i to za razna vremenska razdoblja. Naime, bilo bi neprikladno da se za jedno (npr. agencijsko) rezerviranje otvara toliko rezervacija (tj., da se toliko puta ponavlja pisanje "zaglavlja") koliko se (tipova) smještajnih jedinica rezervira. Stoga, objektu REZERVacija pridružujemo slabi objekt STAVKA, kako je to učinjeno na slici 11.



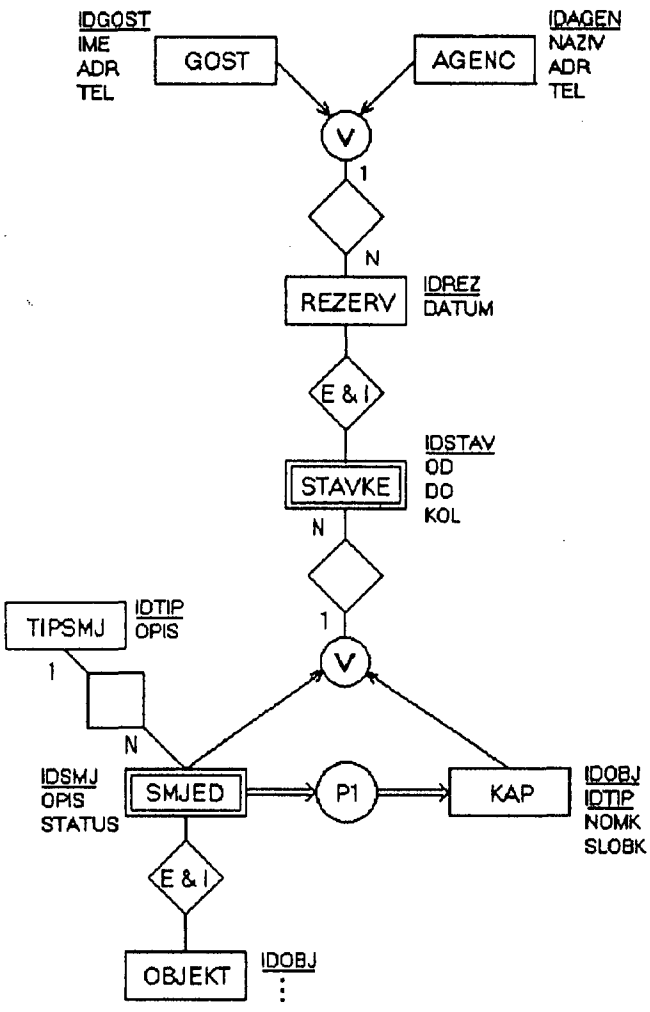
Slika 11

Očito, tako definirani objekti imaju niz svojstava koja se *međusobno isključuju*. Naprimjer, ako rezervaciju vrši agencija, onda je svojstvo IME isključeno ("neprimjenljivo"); jednako neprimjenljivima bivaju svojstva IDAGEN ("šifra" agencije) i NAZIV kada rezerviranje vrši individualni gost. Analogno vrijedi za STAVKE: ako se rezervira tip (IDTIP), onda je broj smještajne jedinice (IDSMJ) neprimjenljiv. U suprotnom, rezervacija točno određene sobe isključuje *potrebu* po eksplicitanju tipa. U takvim situacijama - prema načelu oblikovanja modela podataka - uvode se *podtipovi* (primjereni (disjunktnim) podskupovima instanci datog tipa entiteta). Slijedeći rigorozno to načelo, sačinili bismo model podataka dat na slici 12.



Slika 12

No, takav model ne bi se nipošto odlikovao preglednošću. Nadalje, (nepotrebno) veliki broj relacija (tabela) koje bi iz njega nastale ne bi predstavljao optimalno rješenje ni sa procesnog, a ni sa čisto operativnog (praktičkog) aspekta. Stoga, ovdje predlažemo pojednostavljenje modela (i njegova prikaza!) uvođenjem (simbola) *disjunkcije* na razini veze, kako je to pokazano na slici 13.



Slika 13

Prijevod takova ER modela glasio bi:

GOST(IDGOST,IME,ADR,TEL)
AGENC(IDAGEN,NAZIV,ADR,TEL)
RERERV(IDREZ,DATUM,IDAGEN,IDGOST)
STAVKE(IDREZ,IDSTAV,IDOBJ,IDTIP,
IDSMJ,OD,DO,KOL)
SMJED(IDOBJ,IDSMJ,OPIS,STATUS)
KAPAC(IDOBJ,IDTIP,NOMK,SLOBK)
TIPSMJ(IDTIP,OPIS)
OBJEKT(IDOBJ, ...)

Promatrano sasma operativno, prilikom otvaranja rezervacije, mogao bi biti uspostavljen neki fiktivni indentifikator (npr. redni broj rezervacije) jer taj identifikator nema neke izravne praktičke primjene. Nadalje, nakon preuzimanja datuma, ako se radi o individualnom gostu onda se IDAGEN "preskače" (neprimjenljivo svojstvo), te se uzimaju podaci o gostu. (Način i razlozi (u)vođenja i (trajnog) čuvanja zapisa o (potencijalnom) gostu ovdje su nevažni). U suprotnom, postupa se analogno sa agencijom. (Napomenimo, da se iz "praktičkih razloga" - pored IDAGEN / IDGOST - u rezervaciju, u pravilu, zapisuju i redundantna svojstva poput naziv/ime, adresa, telefon, ...)

Slijedi upisivanje (jedne ili više) stavki, pri čemu IDSTAV može biti jednostavno redni broj stavke u rezervaciji. Ako se vrši rezervacija na razini kapaciteta, onda broj sobe (IDSMJ) ostaje "prazan"; u suprotnom, ispunjavaju se sva polja, s time da KOLičina biva "1".

Dakle, čini se da uvođenje disjunkcije veza s jedne strane daje znatno pregledniji model poda- taka, a s druge strane (implicitno) odražava i sam proces vršenja rezervacije. Naravno, to nipošto ne znači da model podataka "nameće" detalje same *procedure* korištenja informacijskog sistema (npr. redoslijed operacija kod rezerviranja). Procedura rezerviranja može se zacijelo odvijati i drukčije (npr. pitanje što se i gdje rezervira može prethoditi pitanju tko rezervira), pri čemu model podataka ostaje, naravno, sasma nezavisnim od operativnih postupaka takove vrste.

Naravno, specifičnosti mjesta na kojem se vrši rezerviranje (npr. centralna recepcija, ili pak recepcija pojedinog objekta (hotela)) mogu zahtijevati prilagodbu modela. Naprimjer, ako se rezervacija može vršiti samo za jedan objekt (na recepciji tog objekta), onda podaci o objektu zacjelo neće ulaziti u stavke. U tom slučaju, OBJEKT bi - prema potrebi - mogao biti "vezan" na "zaglavlje rezervacije" (tj. na objekt REZERV).

5. Zaključak

Poput svakog formalnog sistema, jezik za oblikovanje modela (baze) podataka "kreće se" između dvaju - u pravilu, protustavljenih - zahtjeva: *po jednostavnosti* i *po "izražajnosti"* (tj. po "semantičkom bogatstvu"). U ovom članku ukazali smo na potrebu i opravdanost proširenja

ER jezika uvođenjem dvaju novih simbola: *simbola procesa* i *simbola disjunkcije*. Pokazali smo kako ti simboli povećavaju mogućnosti (a i jednostavnost!) oblikovanja modela podataka pomoću ER jezika. Naravno, svjesni smo da uvođenje novih elemenata (posebno *procesnih*) može "zasjeniti" sam model *podataka*. No, mnijenja smo da ovdje iznijeti prijedlozi to ne čine, već da prije doprinose njegovoj preglednosti.

REFERENCE

<Bra 87> Brackett, H.M.:
Developing Data Structured Databases, Prentice-Hall, 1987.

<Che 76> Chen, P.P:
The Entity-Relatinship Model: Toward a Unified View of Data, *ACM Trans. on Database Systems*, No. 1, 1976.

<Dat 90> Date, C.J.:
An Introduction to Database Systems, Vol. 1, Addison-Wesley, 1990.

<Fur 86> Furtado, L.A., Neuhold, J.E.:
Formal Techniques for Data Base Design, Springer-Verlag, 1986.

<Haw 88> Hawryszkiewycz, I.T.:
Systems Analysis and Design, Prentice Hall, 1988.

<Mar 87> Martin, J.:
Recommanded Diagramming Standards for Analysts & Programmers: A Basis for Automation, Prentice-Hall, 1987.

<Nav 86> Navathe, S,, Elmasri, R., Larson, J.:
Integrating Users Views in Database Design, *IEEE Computers*, No. 1 1986.

<Per 89> Perry, T.J., Lateer, G.J.:
Understanding Oracle, Sybex, 1989.

<Rad 89> Radovan, M.:
Modeliranje podataka: ER jezik i normalne forme, *Informatica*, No. 1, 1989.

<Rad 91> Radovan, M.:
Projektiranje informacijskih sistema, Informator, 1991.

<Teo 86> Teorey, J.T., Yang, D., Fry, P.J.:
A Logical Design Methodology for Relational Databases Using the Extended Entity-Relationship Model, *ACM Computimg Surveys*, No. 2, 1986.

<Tho 89> Thompson, J.P.:
Data with Semantics, Van Nostrand Reinhold, 1989.

<Ull 88> Ullman, D.J.:
Principles of Database and Knowledge-base Systems, Vol. 1, Comp. Sci. Press, 1988.