MS Access

(1) OSNOVE UPRAVLJANJA RELACIONIM BAZAMA PODATAKA

R.S. Mikanović

Microsoft Office 2000

Sadržaj

Pre svega,

dobro došli u svet lakog i, iznad svega, zanimljivog ostvarenja programerskih ideja. Svaki program zahteva obradu podataka, pa krenimo baš od toga – BAZE PODATAKA i naučimo to na najbolji način.

Kurs *MS Access –(1) Osnove upravljanja relacionim bazama podataka* je osmišljen kao jedan oblik ulaska u ovu kompleksnu materiju, ali na lakši način.

Da bi mogao da prati kurs na efikasan način, od polaznika se zahteva da ima ozbiljno iskustvo u obradi teksta (*MS Word*), kao i u obradi tabela (*MS Excel*), a pohađanje ovog kursa se savetuje:

- korisnicima razvijenih aplikacija pod *MS Access*-om,
  jer će naučiti standard u korišćenju *MS Access* baza podataka i informacionih sistema
- programerima koji žele da u svoje buduće projekte uključe baze podataka, jer će naučiti da kvalitetno upravljaju onim objektima koji su i najznačajniji za kvalitetnu bazu podataka (tabele i upiti)
- onima koji žele da se profesionalno bave *MS Access*-om,
  jer će naučiti kako se jednostavno mogu dizajnirati obrasci i izveštaji, a edukaciju mogu nastaviti kroz više kurseve *MS Access*-a.

U toku trajanja kursa polaznici bi trebalo da ovladaju osnovnim tehnikama korišćenja *MS Access*-a, ali i kreiranja jednostavnih relacionih baza podataka. Pri radu polaznici mogu odabrati jednu od verzija *MS Access*-a verzija 8.0 iz paketa MS Office 97 - Professional Edition ili verzija 9.0 Microsoft Office 2000- Premium. Kroz teme kursa obrađuje se relacioni model baza podataka, što znači da će se izučavati objekti baze i veze između objekata (relacije).

Radi lakšeg razumevanja tematike Kurs je podeljen na grupe tema A) Osnovni pojmovi, principi i standard relacionih baza podataka (gde će biti prikazano nekoliko primera edukativnih i poslovnih baza podataka), B) zatim malo teorije, C) radno okruženje *MS Access*-a, organizacija tabela, njihovo kreiranje i dizajniranje, D) upiti (*Queries*) i SQL, E) osnove kreiranja obrazaca (*Forms*) i izveštaja (*Reports*).

Kako su najvažniji objekti baze podataka tabele (*Tables*) i upiti (*Queries*), od njihovog kvalitetnog kreiranja zavise svi drugi objekti u bazi, pa ćemo tako i mi težište našeg rada na ovom Kursu baciti baš na ove teme. Dizajn obrazaca (*Forms*) i izveštaja (*Reports*) u *MS Access*-u traži mnogo više vremena, pa će se na Kursu raditi samo osnove dizajna ovih objekata, a detaljna obrada ovih tema je predviđena na *MS Access* (2).

Zašto, za projektovanje baza podataka, koristimo baš *MS Access*? Pored mnogih razloga vredi pomenuti: jednostavnost rada i lakoću ostvarenja zamišljenog, standardno *Windows* grafičko okruženje, lako dostupan softver, mogućnost korišćenja starijih modela baza podataka radi njegovog daljeg razvoja,...

Tokom kursa predviđen je i praktični rad u okviru kog ćemo zajednički kreirati bazu podataka "Kadrovi" i kroz njen razvoj učiti jednu po jednu temu, dok ćemo upite savladati kroz urađene primere u bazi "Magaza" sa unetim podacima. Da biste sagledali sve aspekte, primetili razlike i otkrili greške, mnogi primeri će biti prikazani u neispravljenom (sirovom) obliku, što je naročiti pogodno za provociranje grešaka i/ili korigovanje primera.

Od polovine kursa, pa sve do poslednjeg termina, vaš će zadatak biti da, na osnovu jednog jednostavnog (šematski prikazanog) relacionog modela, kreirate bazu podataka po svojim idejama i prema svojim potrebama, učemu ćete imati punu podršku predavača. Na kraju Kursa, Vaš zadatak će biti da, unosom eksperimentalnih podataka, prezentirate kolegama u grupi – Vašu bazu podataka, pa da zajednički sagledamo moguće puteve njenog daljeg razvoja.

Oblast projektovanja relacionih baza podataka, razvoj aplikacija i informacionih sistema pod *MS Access*-om, gde je potrebno šire znanje, ali i iskustvo u radu, se izučava na višim kursevima i to:

MS Access (2) - Osnovne tehnike razvoja aplikacija
gde se obrađuje optimizacija baza podataka, dizajn formi i izveštaja, i

MS Access (3) - Razvoj Informacionih Sistema,
gde se polaznici obučavaju za poslove projektovanja, razvoja i održavanja IS-a.

Po završetku ova (1) i (2) stepena edukacije i uz vaše permanentno zalaganje bićete sposobni da izvršite sve zadatke u oblasti konkretne primene *MS Access*-a u projektovanju poslovnih baza podataka.

Svi potrebni primeri koji su ovde pomenuti mogu se preuzeti od autora Mikac@BitsYu.net ili 064/1168683.

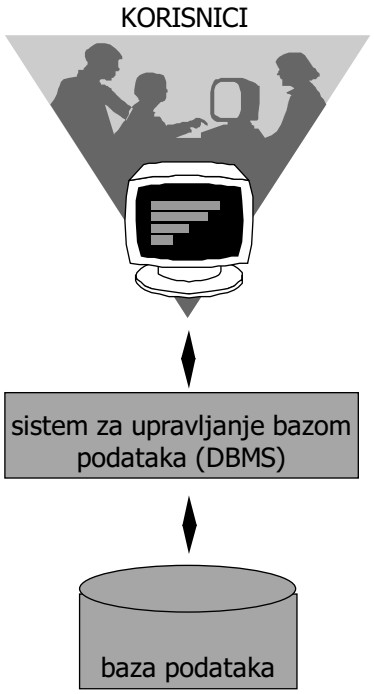
A u t o r

1 OSNOVNI POJMOVI I STANDARD

Podatak - nosilac realne slike stvarnosti koji daje opis nekog činjeničnog stanja (žut; beo; ceo; 1; F14; 354; $10,50; Car Dušan; itd.). To je, u stvari, činjenica o nekom segmentu realnog sveta koju treba čuvati u elektronskom obliku na duži vremenski period i koja ima značenje za korisnike baze podataka.

Informacija – obrađeni podaci povezanih grupa podataka ili podatak koji proširuje nivo znanja, odnosno smanjuje entropiju sistema (neodređenost). Za informaciju je bitno da bude jasna, tačna i pravovremena. Tako bi rečenica: "Danas je naša firma ostvarila promet u iznosu od $5000 sa zaradom od $320" bila informacija, ako nas obaveštava o nečem što nismo znali.

Baza podataka - kolekcija povezanih podataka sa višestrukom namenom. To je skup podataka koji se odnose na jednu temu ili namenu. Možda je lakše shvatiti bazu podataka kao skladište opšte namene za smeštaj i obradu bilo koje vrste informacija, prema vašoj želji. Na primer, u bazu podataka možete smeštati imena i adrese. Takođe biste mogli da skladištite podatke o vašoj videoteci, fonoteci, zalihama i porudžbinama vašeg privatnog preduzeća, kao i svih oblika evidentiranja i prikazivanja podataka o poslovanju.



Baza podataka je više od samih podataka. Ona sadrži objekte koji će vam pomoći pri upravljanju podacima, kao što su obrasci (za unos ili ažuriranje podataka) i izveštaji (za prikaz i/ili štampanje probranih grupa podataka u odgovarajućem obliku) kao i jedna ili više tabela u koje skladištite podatke koji su značajni za vas.

Sistem baze podataka – je celina koju čine:

- sama baza podataka,
  koja može predstavljati jednu datoteku, dok je to u kompleksnijim modelima najčešće hijerarhijski i strukturno organizovan sistem više povezanih datoteka
- sistem za upravljanje bazom podataka
  (Database Management System - DBMS) i
- korisnici baze podataka.

Na slici desno je predstavljen opšti model sistema.

MS Access je kompleksan skup više zavisnih alata za kreiranje i razvoj baza podataka, pa je teško reći da je to tek jedan program, ali se u svakom slučaju može svrstati u kategoriju visoko razvijenih DBMS razvojnih alata (programa).

DBMS je jedan složen softverski alat koji treba da omogući:

- **Skladištenje** podataka sa minimalnom redudansom (ponavljanje istih podataka) i njihovu centralizaciju, čime se postiže jednostavna i efikasna organizacija podataka.
- **Korišćenje** zajedničkih podataka od strane svih ovlašćenih korisnika, kao i podelu korisnika sistema na grupe prema pravima pristupa.
- **Logičku i fizičku nezavisnost programa** od podataka. Bez obzira što se podaci fizički pamte, po pravilu, samo jednom, u jedinstvenoj fizičkoj organizaciji, svaki korisnik dobija svoju sopstvenu logičku sliku (logička struktura) podataka kakva mu najviše odgovara.
- **Jednostavno korišćenje** baze podataka preko grafičkog interfejsa bliskog korisniku, čime se krajnji korisnici lakše uključuju u razvoj informacionog sistema (IS), pa time njegovo korišćenje uglavnom ne predstavlja nikakav problem. Licima koja učestvuju u razvoju ovakvog sistema značajno se povećava produktivnost, odnosno rezultati se postižu za relativno kratko vreme.
- **Najracionalnije iskorišćenje** zauzetog prostora na spoljnoj memorijskoj jedinici.

Informacioni sistem - je visokouređeni skup metoda, procesa, i operacija za prikupljanje, čuvanje, obradu, prenošenje i distribuciju podataka u okviru jedne organizacije, uključujući i opremu koja se u te svrhe koristi i ljude koji se tim aktivnostima bave. Kao primeri mogu poslužiti informacioni sistemi poslovnih banaka i svih većih trgovinskih preduzeća, spoljno-trgovinskih preduzeća , projektnih biroa, avio kompanija i slično.

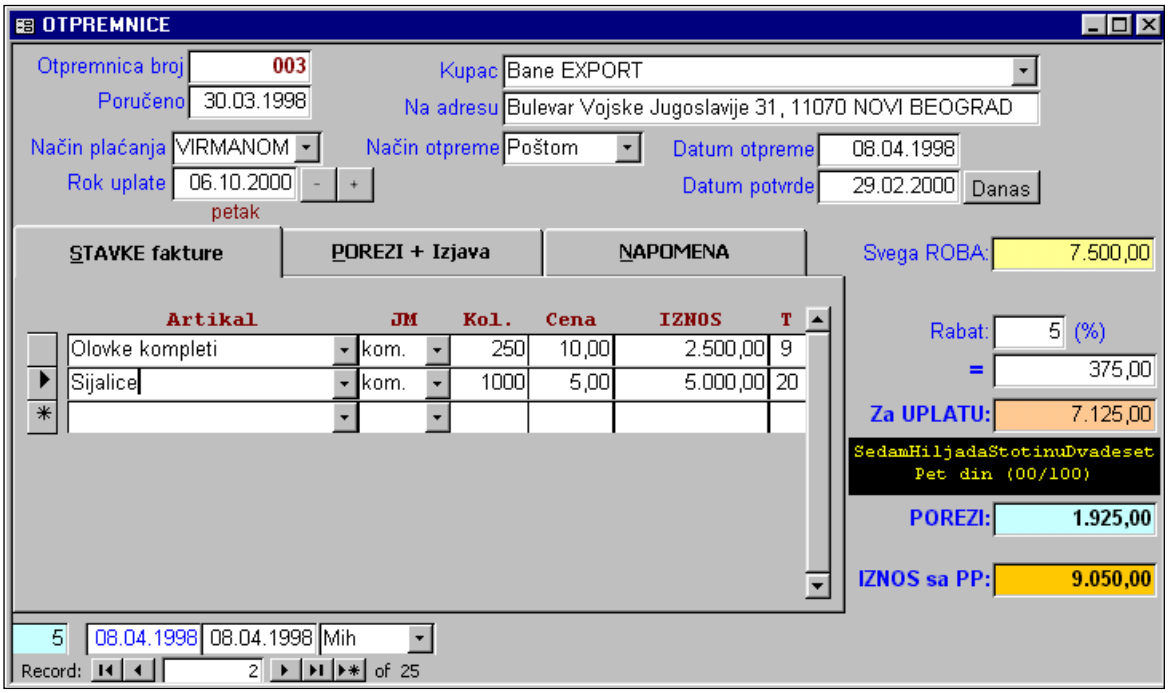
Opšta arhitektura sistema za upravljanje bazom podataka koja omogućuje ostvarivanje navedenih ciljeva sastoji se iz tri nivoa:

Interni fizički nivo definiše način na koji su podaci fizički organizovani na spoljnim memorijskim jedinicama (diskovima).

Konceptualni nivo definiše opštu logičku strukturu baze podataka, sve podatke u modelu i njihove logičke odnose (veze), treba da omogući upravljanje podacima kao zajedničkim resursom u sistemu baze podataka.

Eksterni nivo (korisnički) na njemu se definiše logička struktura podataka pogodna za specifične zahteve, odnosno programe.

Iako su tabele primarni izvori podataka, korisnici baze podataka nemaju direktan pristup ovim objektima, već obrađi podataka pristupaju preko određenih obrazaca ili formulara (*Forms*) za unos i ažuriranje podataka, koji se nalaze u bazama koje mogu da predstavljaju aplikacije pod *Access*-om. Grafičke mogućnosti rada pruža operativni sistem *Windows*, a nazivamo je grafički korisnički interfejs (GUI).



Na slici predstavljen je jedan primer obrasca (*Form*) u grafičkom interfejsu, a koji se koristi za ažuriranje podataka o otpremnicama i njihovim stavkama.

Korisnici i osoblje koje učestvuje u projektovanju i razvoju baze može se podeliti u nekoliko grupa:

Izvršioci projektanti baze podataka, administrator baze podataka, sistem analitičari, aplikativni programeri, kao i nekoliko lica angažovanih od strane investitora i krajnjih korisnika sistema.

Podrška projektanti i programeri DBMS-a, inženjeri u razvoju zaduženi za softverske alate i osoblje zaduženo za razvoj softvera koje obuhvata sistem baze podataka, hardvera i

Krajnji korisnici stalni, povremeni i slučajni korisnici sa različitim stepenom iskustva i prioritetom u pristupu podacima.

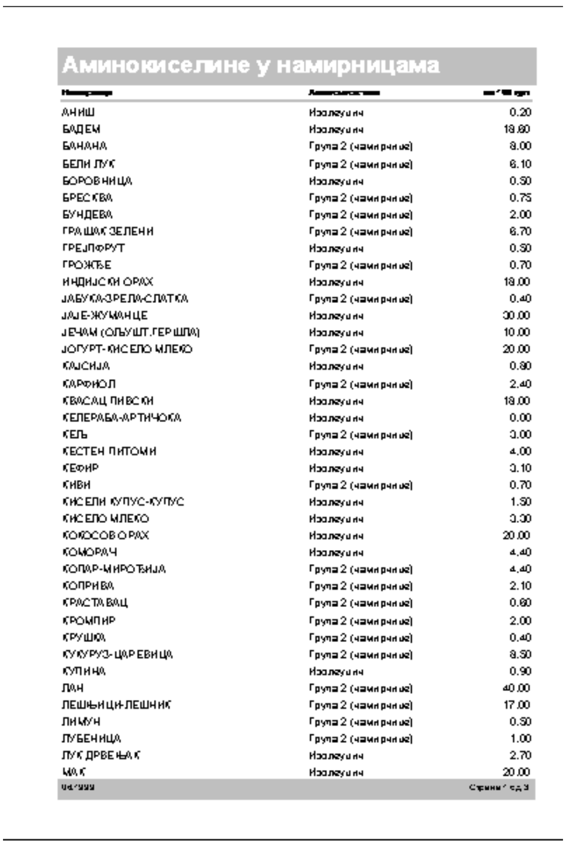
Standard u projektovanju baza podataka

Da bi baza podataka bila smatrana standardnom mora da ispunjava najmanje sledeće uslove:

- da se tekstualni podaci unose prema kodnim stranama: 1250, 1251 ili 1252 (YU tastatura, latinična ili ćirilična za naš region), koje ne prave razliku između velikih i malih slova;
- da se za format ispisa svih (numeričkih, valutnih, datumskih, vremenskih,…) podataka oslanja na podešavanja na nivou OS-a (u *Regional Settings*) i/ili korisničkim opcijama u aplikaciji), a u potpunosti mora da odgovara korisniku;
- da se za veličinu polja u tabelama uzimaju racionalne dužine, prema opsegu i veličini samog podatka u tom polju;
- da ima mrežnu orijentaciju, zaštićen relacioni model i zahtevani referencijalni integritet;
- da ima neki oblik standardnog sistema menija;
- da se lako administrira, može dalje razvijati i ima period korišćenja duži od jednog poslovnog perioda (uglavnom 6 ili 12 meseci);
- da, po isteku test perioda, ne zavisi od autora, odnosno da je može administrirati liceobučeno za takve poslove u firmi korisnika;
- da korisniku pruža na raspolaganje sistem pomoći, arhiviranje rezervne kopije;
- da je urađena po projektnoj dokumentaciji (u slučaju IS-a) i da su sve izmene u modelu baze - dokumentovane.

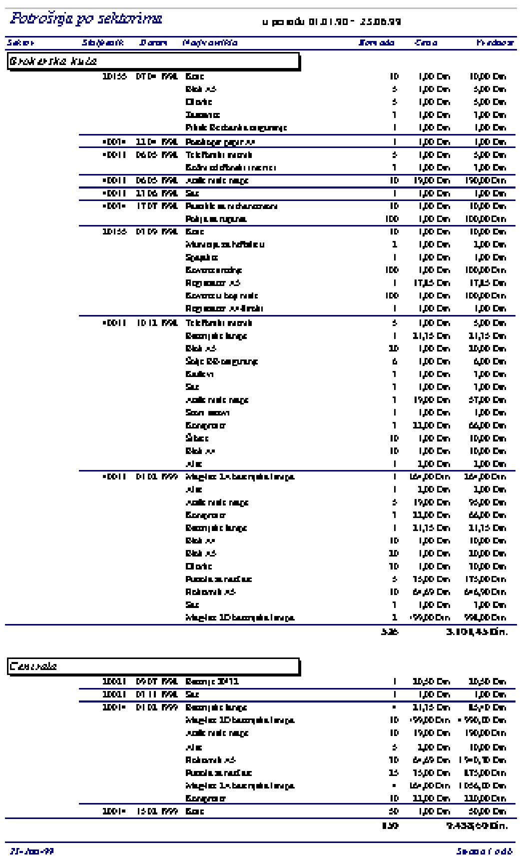
Poštovanje standarda u razvoju baze podataka je od izuzetnog značaja, a ono što bi bilo dovoljno da vas usmeri ka tom cilju je:

- jednostavnost korišćenja i lakša implementacija,
- olakšane naknadne izmene i usklađivanje po zahtevu i potrebama korisnika,
- bezbednost podataka i veliki uticaj korisnika na oblik (format) njihovog prikaza,
- totalna korisnička orijentacija, što znači da korisnik baze podataka, odgovore na sva pitanja o korišćenju, dobija od same baze (aplikacije).



Аминокиселине у намирницама

АНИШ	Изолеуцин	0.20
БАДЕМ	Изолеуцин	18.80
БАНАНА	Група 2 (намирнице)	8.00
БЕЛИ ЛУК	Група 2 (намирнице)	6.10
БОРОВНИЦА	Изолеуцин	0.50
БРЕСКВА	Група 2 (намирнице)	0.75
БУНДЕВА	Група 2 (намирнице)	2.00
ГРАШАК ЗЕЛЕНИ	Група 2 (намирнице)	6.70
ГРЕЈПФРУТ	Изолеуцин	0.50
ГРОЖЂЕ	Група 2 (намирнице)	0.70
ИНДИЈСКИ ОРАХ	Изолеуцин	18.00
ЈАБУКА-ЗРЕЛА-СЛАТКА	Група 2 (намирнице)	0.40
ЈАЈЕ-ЖУМАНЦЕ	Изолеуцин	30.00
ЈЕЧАМ (ОЉУШТ.ГЕРШЛА)	Изолеуцин	10.00
ЈОГУРТ-КИСЕЛО МЛЕКО	Група 2 (намирнице)	20.00
КАЈСИЈА	Изолеуцин	0.80
КАРФИОЛ	Група 2 (намирнице)	2.40
КВАСАЦ ПИВСКИ	Изолеуцин	18.00
КЕЛЕРАБА-АРТИЧОКА	Изолеуцин	0.00
КЕЉ	Група 2 (намирнице)	3.00
КЕСТЕН ПИТОМИ	Изолеуцин	4.00
КЕФИР	Изолеуцин	3.10
КИВИ	Група 2 (намирнице)	0.70
КИСЕЛИ КУПУС-КУПУС	Изолеуцин	1.50
КИСЕЛО МЛЕКО	Изолеуцин	3.30
КОКОСОВ ОРАХ	Изолеуцин	20.00
КОМОРАЧ	Изолеуцин	4.40
КОПАР-МИРОЂИЈА	Група 2 (намирнице)	4.40
КОПРИВА	Група 2 (намирнице)	2.10
КРАСТАВАЦ	Група 2 (намирнице)	0.60
КРОМПИР	Група 2 (намирнице)	2.00
КРУШКА	Група 2 (намирнице)	0.40
КУКУРУЗ-ЦАРЕВИЦА	Група 2 (намирнице)	8.50
КУПИНА	Изолеуцин	0.90
ЛАН	Група 2 (намирнице)	40.00
ЛЕШЊИЦИ-ЛЕШНИК	Група 2 (намирнице)	17.00
ЛИМУН	Група 2 (намирнице)	0.50
ЛУБЕНИЦА	Група 2 (намирнице)	1.00
ЛУК ДРВЕЊАК	Изолеуцин	2.70
МАК	Изолеуцин	20.00



Nekoliko standardnih baza podataka (aplikacija) ćete videtiu prezentaciji na prvom predavanju.

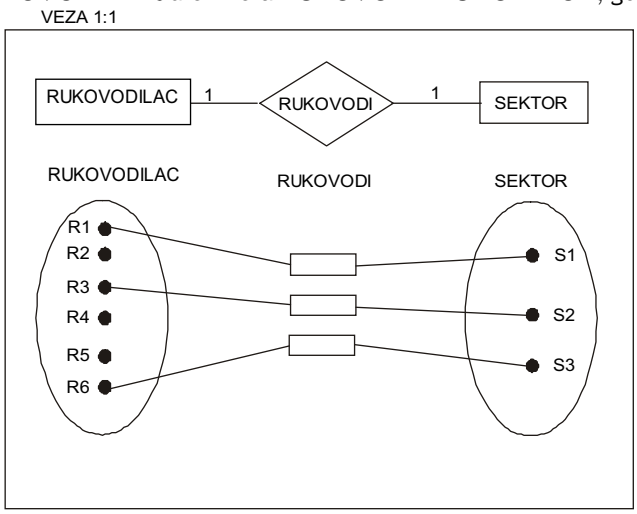
2 ENTITETI I VEZE MEĐU NJIMA

Entitet je element o kome se memorišu informacije. To je element koji postoji i koji se može razlikovati od ostalih elemenata. Entitet predstavlja bilo šta o čemu se mogu memorisati opisne informacije, što je u stanju da postoji nezavisno i što može biti jednoznačno određeno. Entitet može biti AUTOMOBIL, a njegovi atributi: marka, tip, boja, godina proizvodnje, i sl. Podaci u okviru baze nisu raspoređeni haotično, nego su grupisani po entitetima, što znači da svaki entitet sa sobom nosi određenu vrstu podataka.

Jedna baza se može sastojati od jednog ili više entiteta. Između entiteta mogu postojati određene veze koje su ilustrovane u sledećim primerima i mogu biti sledećih oblika:

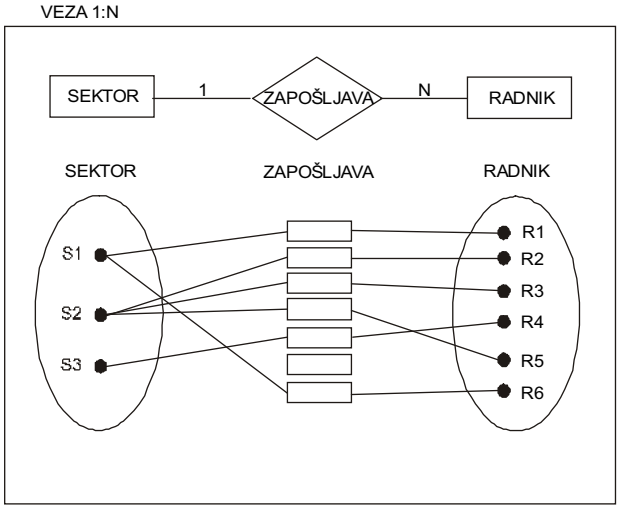
Veza 1:1

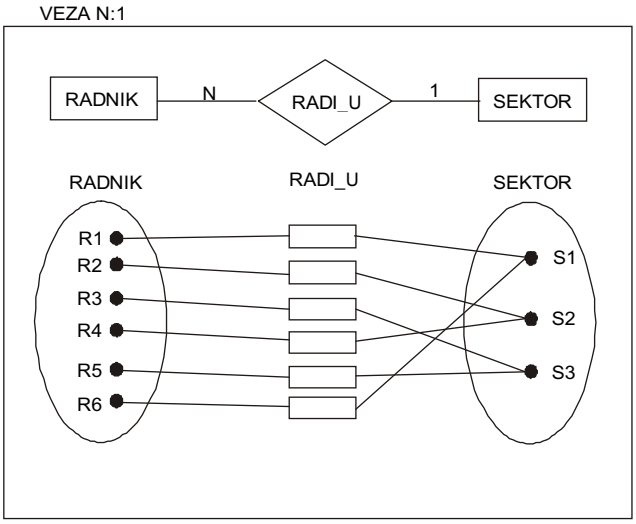
1:1 je najprostiji oblik veze, a za primer se može uzeti veza RUKOVODI između entiteta RUKOVODILAC i SEKTOR, gde se vidi da jedan rukovodilac može rukovoditi samo jednim sektorom, ali i obratno - jedan sektor ima samo jednog rukovodioca.



Veza 1:n (n:1)

je najčešći tip veze, a da li je veza 1:n ili n:1 zavisi od toga u kom smeru je posmatramo. Za primer ovog tipa veze se može uzeti veza SEKTOR ZAPOŠLJAVA RADNIKA, gde su entiteti SEKTOR i RADNIK u vezi 1:n, jer u jednom sektoru može biti zaposleno više radnika.



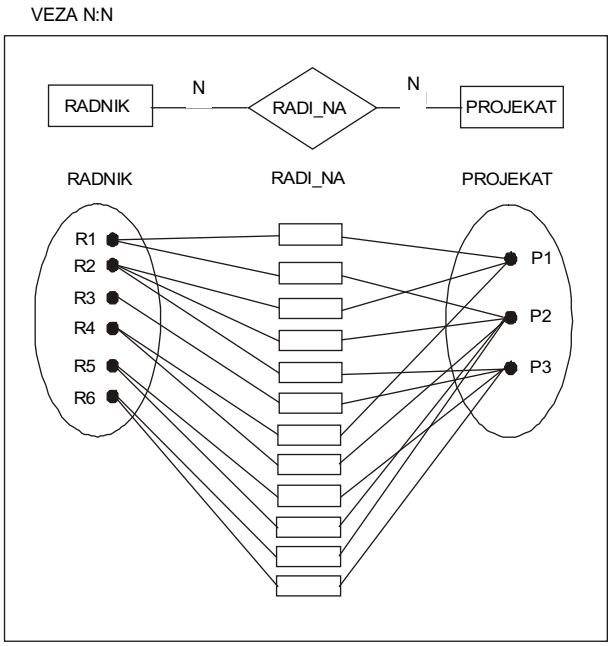


Posmatrajući ovu istu vezu u suprotnom smeru možemo učiti vezu RADNIK RADI_U SEKTORu, gde više radnika može raditi u jednom sektoru.

Veza n:n

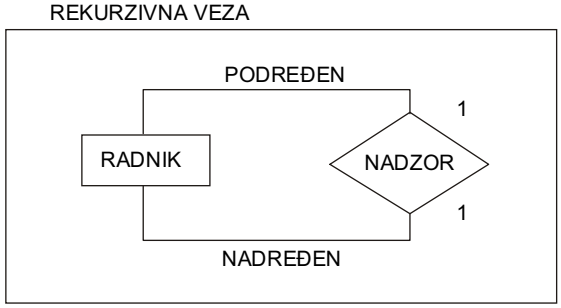
je veza (više prema više) koja se u modelima vrlo često javlja, a ono što je za nju specifično je da takva kakva jeste - ne može da se direktno implementira u relacionom modelu baze podataka, jer bi dovela do nemogućnosti stroge definicije.

Problem veze n:n između dva entiteta se prevazilazi "razbijanjem" ove veze na dve veze tipa 1:n, pa se tako, prikazana veza RADNIK:PROJEKAT (n:n), deli na dve veze: PROJEKAT:P_R (1:n) i P_R:RADNIK (n:1), gde je P_R novi entitet (tabela u bazi) koji obezbeđuje strogu kontrolu modela.



Rekurzivna veza

još jedan problematičan tip veze, a sreće se u situacijama kada se između dva (ili više) entiteta pojavi veza 1:n prema jednom paru atributa, a 1:1 ili n:1 prema drugom paru. Kao primer možemo uzeti vezu RADNIK:RUKOVODILAC.



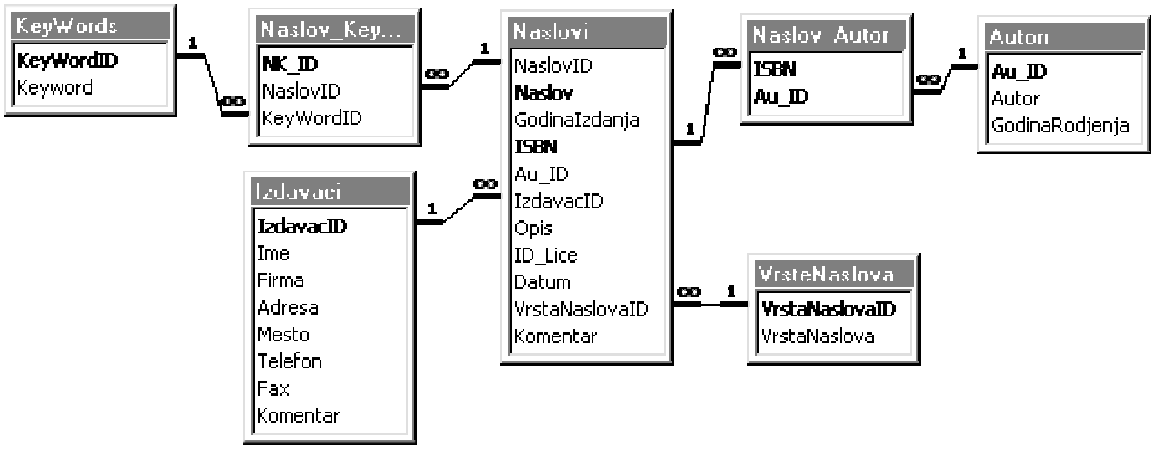
Jedan radnik se, u tabeli RADNICI, može pojaviti samo jednom, dok se kod izbora nadzornika (rukovodioca) u tabeli RUKOVODIOCI, jedan radnik opet može pojaviti samo jednom, pa se uočava veza 1:1, ali pošto je jedan rukovodilac nadređen većem broju radnika, jasno je da postoji i veza RUKOVODIOCI:RADNICI = 1:n. Objedinjavajući ove dve veze u jednu relaciju - proizlazi da je ona tipa n:n.

Ovaj tip veze je u bazi podataka moguće ostvariti, ali se model teško može održavati, pa se savetuje njeno obavezno eliminisanje, tj razbijanje na više entiteta sa vezama 1:n.

Primer povezanosti entiteta jedne baze podataka

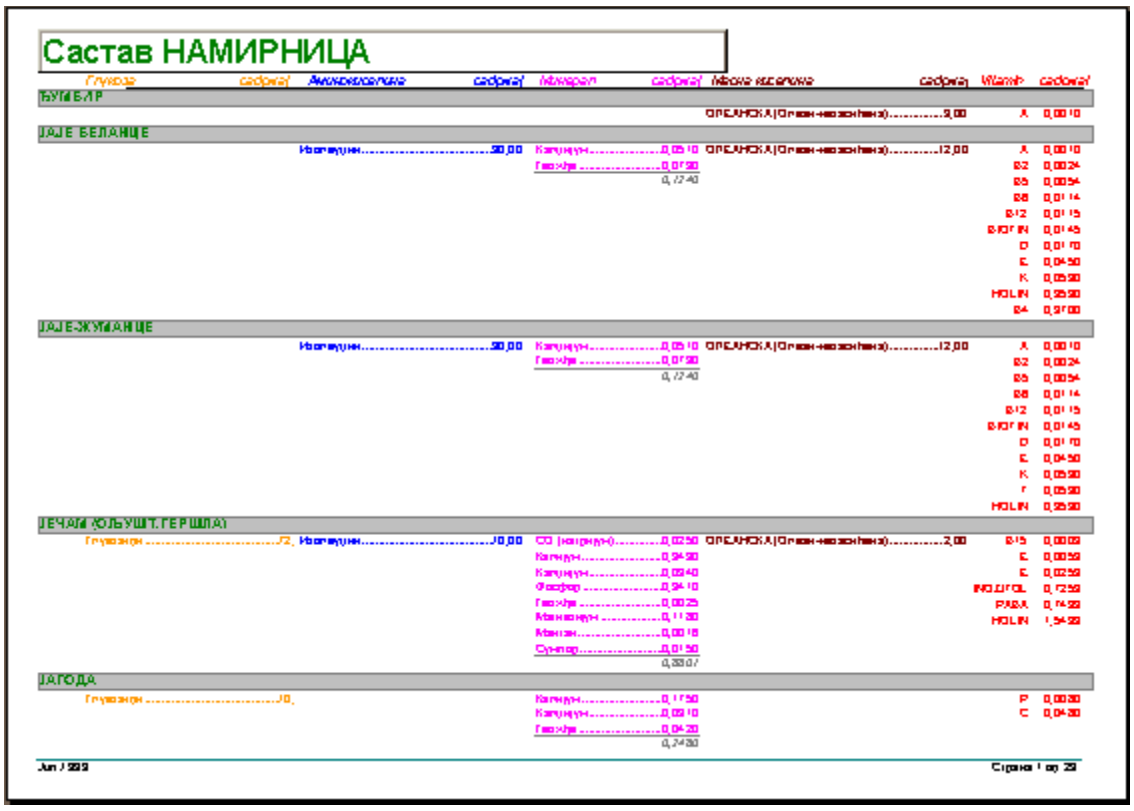
Na sledećoj slici su predstavljeni entiteti baze podataka *Biblios* (biblios.mdb) i veze među njima. Na Web adresi http://www.interbest.co.yu/source/access/ možete preuzeti (u ZIP formatu) neke od primera baza podataka koji su u skriptama izloženi, test pitanja, dodatke, index pojmova i slično, ili se obratite autoru na Mikac@BitsYu.net.

Entiteti u ovoj bazi su: Naslovi, Autori, Izdavaci, Naslov_Keywords, Naslov_Autor, Lica i VrsteNaslova.



Na slici se vidi da se izdavač u tabeli IZDAVACI može naći samo jednom, a u tabeli NASLOVI više puta. Naslov može imati jednog izdavača, a jedan izdavač se može naći u više naslova, pa je veza NASLOVI:IZDAVACI, preko polja (atributa) IzdavacID, tipa n:1 (∞:1).

Veza NASLOVI:AUTORI je tipa n:n (na istom naslovu se može naći više autora, a sam autor može učestvovati na više naslova), pa je dodatnom tabelom NASLOVI_AUTORI obezbeđeno njeno "razbijanje" na dve veze tipa 1:n.



3 UVOD U MS ACCESS

Objekti MS Access-a

MS Access spada u grupu programa za upravljanje bazama podataka - DBMS (*Database Management System*). Kao što i samo ime kaže, ovi alati se koriste u projektovanju, razvoju i korišćenju baza podataka. Podaci koji se obrađuju mogu praktično biti sve što je, za korisnika baze podataka, prirodno i potrebno:

- Imena i adrese
- Informacije o zaposlenima i saradnicima
- Fakture, uplate, knjigovodstvo
- Poslovni kontakti, kupci, predviđanja prodaje
- Podaci o kretanjima i stanjima zalihama
- Planovi, rezervacije, projekti itd.

PRIMER: Pogledati kako su podaci organizovani u demo bazi MAGAZA (magaza.mdb u arhivi magaza.zip) ili neku od baza koje ste dobili uz instalaciju *MS Access*-a.

Jedna baza podataka u *MS Access*-u može da sadrži sledeće vrste objekata:

Tabela (*Table*) je osnovni tip objekta u bazi i predstavljaju direktan (primarni) izvor podataka. U tabelama se, po osmišljenim principima i vezama, čuvaju podaci kojima raspolaže korisnik i one su prvi objekti koje treba kreirati. Podaci u tabeli su smešteni u polja (kolone, eng. *Fields*), a sva definisana polja čine slog (zapis, red, eng. *Record*). Kvalitet baze podataka i IS-a leži u kvalitetnoj organizaciji podataka u tabelama baze, kao i njihovim dobrim vezama (relacionom modelu).

Upit (*Query*) je tip objekta za postavljanje pitanja o podacima iz tabela (ili drugih upita), a u cilju njihovog ažuriranja kroz obrasce ili pregleda kroz izveštaje (na ekranu ili štampaču), pa se mogu definisati kao posredni izvori podataka. Na primer, upiti mogu da daju informacije (odgovore na pitanja) kao što su "koliko kupaca je iz Novog Sada, koja su njihova imena i brojevi telefona", s tom razlikom što vaš nadobudavac govori srpski, a upit - jezikom *SQL*. Upiti se mogu koristiti za spajanje (engl. *join*) kolona iz više (relacijama povezanih), tabela. Jedan upit bi mogao, na primer, da spoji tabele kupaca, porudžbina, stavki porudžbina i proizvoda da bi odgovorio na pitanje "koji kupci su naručili koje proizvode i koja je vrednost njihovih porudžbina". Upiti su, takođe, korisni prilikom izmene, brisanja ili unošenja velike količine podataka u jednom prolazu. Svaki korisnik *MS Access*-a bi trebalo da ga bar toliko poznaje da može bez problema iz njegove baze "izvući" potrebne podatke i to onako kako mu zada kolega ili nadobudavac. Svaki dalji rad u upravljanju bazom se zasniva na kreiranju objekata (formei izveštaji) koji ne mogu bez upita ili tabela, pa se njihovo kreiranje može znatno ubrzati samo pod uslovom da su izvori podataka (tabele i upiti) dobro kreirani.

Obrazac (*Form*) omogućava unos i prikazivanje podataka u prikladnom formatu koji liči na štampane obrasce u kojima treba popuniti prazna mesta. Vaši obrasci mogu biti jednostavni ili prilično složeni - sa grafikom, linijama, mogućnostima automatskog pretraživanja, koje unos podataka čine brzim i lakim. Obrasci mogu da sadrže i druge obrasce (nazvane pod-obrasci, engl. *subforms*) što omogućuje istovremeni unos podataka u više tabela.

Izveštaj (*Reports*) daje izvanredne mogućnosti pregleda i štampanja podataka. Kao i obrasci, izveštaji mogu da budu jednostavni, ali i veoma kompleksni. Primeri izveštaja su: spiskovi, hronološki pregled poslovanja, cirkularna pisma, nalepnice sa adresama i fakture. Izveštaji se za podatke obraćaju tabelama, ali još češće upitima, a njihov osnovni zadatak je da te podatke predstave u obliku koji je lak za pregled, razumljiv i gde se mogu brzo uočiti greške. Na primer, izveštajem se može dobiti "prodaja po mestima", "broj porudžbina u nekom periodu" i druge informacije koje se koriste za donošenje važnih poslovnih odluka. Primeri nekoliko izveštaja su prikazani kroz ova skripta (na slobodnim prostorima).

Pages ili *Data Access Pages* (Web stranice za pristup podacima), omogućuju kreiranje složenih obrazaca na intranetu kompanije koji su povezani s podacima u vašoj mreži. Čak i novajlija može da kreira Web stranice sa uvek ažurnim podacima. Hiperveze i vama i drugima omogućavaju pristup podacima koje ste objavili na Webu (i koje su drugi objavili), i to u formatu hipertekstualnih veza, direktno iz vaših *MS Access* obrazaca. Mnogi smatraju da posao objavljivanja podataka na Webu treba da bude prepušten isključivo administratoru Web lokacije (Webmasteru). *MS Access* 2000 je nepobitan dokaz da to nije tačno. Alatka *Data Access Pages* vodi vas kroz postupak izrade obrasca i njegovog povezivanja s podacima u izabranim objektima baze podataka, a zatim generisane HTML strane šalje na vašu Web lokaciju.

Makro (*Macro*) je niz *MS Access* komandi, a koristi se u slučajevima kada nekoliko komandi (određenim redosledom) treba pozvati na više mesta u *MS Access* aplikaciji (uglavnom iz forme). Kada pokrenete makro,

MS Access izvršava sve naredbe makroa u redosledu u kom su akcije navedene. Bez pisanja programskog koda, možete definisati makroe koji automatski otvaraju obrasce za bazu podataka, štampaju nalepnice sa adresama, obrađuju porudžbine i drugo. Makroi olakšavaju sastavljanje skupova tabela upita, obrazaca i izveštaja u kompletne aplikacije koje može da koristi bilo ko,čak i ako zna malo ili ne zna ništa o samom *MS Access*-u. Makroi se izbegavaju u profesionalnim *MS Access* aplikacijama, jer se oni ne mogu kompajlirati, pa ne daju mogućnosti kreiranja aplikacije.

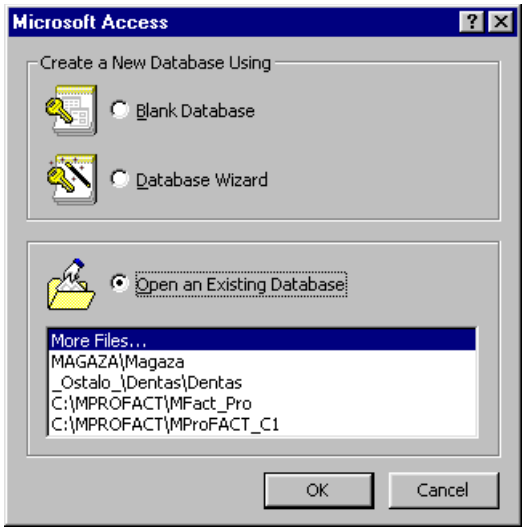
Modul (*Module*) kao i makro omogućava automatizaciju, ali mnogo napredniju, jer se u njemu, kao razvojna platforma, koristi *VBA*, u obliku procedura tipa *Sub* i *Function*, koje su na raspolaganju celom projektu. Za razliku od makroa moduli omogućavaju precizniju kontrolu nad preduzetim akcijama i zahtevaju iskustvo u programiranju u VBA (*Visual Basic for Applications*), a ako uzmemo u obzir da forme i izveštaji imaju takođe modul kôda, jasno je da VBA predstavlja osnovnu programersku razvojnu platformu za *MS Access*. Na ovom kursu se nećemo baviti temama vezanim za VBA.

Kako su tabele izvori podataka, tako *MS Access*-fajl predstavlja bazu u slučaju da ima tabele, dok se u *MS Access* aplikacijama uglavnom nalaze ostali objekti, a veza sa tabelama se uspostavlja *Link*-om. Koncepcija odvajanja baze i aplikacije se obavezno primenjuje u profesionalnim projektima, a šta sve treba uraditi i kakve se prednosti ovog odvajanja - radićemo na višem kursu *MS Access* (2).

OTVARANJE baze iz MS Access-a

Pokretanjem *MS Access*-a, (npr. na *Start/Programs/Microsoft Access)* dobićete dijalog kao na slici i ponuđene sledeće opcije ulaza:

- *Create a New Database Using*
  (kreiranje nove baze koristeći...)
  - *Blank Database*
    (... praznu bazu)
  - *Database Wizard*
    (automatsko generisanje baze)
- *Open an Existing Database*
  (otvaranje postojeće baze podataka)



Database Wizard

Database Wizard pomaže početnicima da na lakši način uđu u rad na bazama i vide nekoliko primera aplikacija Ovim se kreira cela baza za nekoliko minuta, a ponuđeno je više tipova baza podataka (adresari, porudžbine, vođenje inventara, videoteke, fonoteke...). Posle odabrane varijante, odlučujemo se o izboru tabela, i polja u tabelama, uključenju probnih podataka, izgledu aplikacije, izveštaja, i nazivu aplikacije. Sledi automatski proces kreiranja objekata i posle kraćeg vremena- baza je na raspolaganju, kao jednostavna aplikacija koja je jednostavna, ali vrlo korisna za učenje principa rada u *MS Access*-u.

Iako predstavlja dobar način da se vidi kako mogu izgledati aplikacije pod *MS Access*-om, ovaj način ipak nećete mnogo koristiti kod kreiranja svoje baze i aplikacije. Jedan od razloga je i njegovo pravilo da se imena polja u tabelama (koji se dalje prenose na sve ostale objekte) imenuju na engleskom jeziku.

Čarobnjaci u kreiranju svih drugih objekata, osim tabela, sigurno će vam mnogo značiti u *MS Access*, pa ih od srca preporučujemo, a naročito ćemo ih koristiti kod kreiranja formi (obrazaca) i izveštaja. U toku rada u *MS Access*-u preporučuje se što više konsultovanja primera gotovih baza koje *Microsoft* isporučuje uz instalacionu verziju paketa (*Nwind, Orders, Solutions*)

Na prvi pogled

Radni prozor *MS Access* baze (*Database Window*) sadrži *CommandBar* (preko kojeg se otvaraju kartice željenih tipova objekata), kao i svoj *Toolbar*, sa nekoliko najčešćih komandi. U samom vrhu ovog prozora (naslovna traka) nalazi se ime baze koja se obrađuje.

Sistem menija i paleta alatki u *MS Access*-u funkcioniše na sličan način kao i ostalim programima *Microsoft Office*-a, dok se sve ostalo smatra radnom površinom u kojoj se razvija i radi *MS Access* aplikacija. Na samom dnu *MS Access* prozora smeštena je statusna linija (*Status bar*) preko koje se korisniku prosleđuju dodatne informacije o tekućem objektu i informacije koje se odnose na tastere *Insert, Num Lock, Caps Lock, Scroll Lock,...*

Database Window

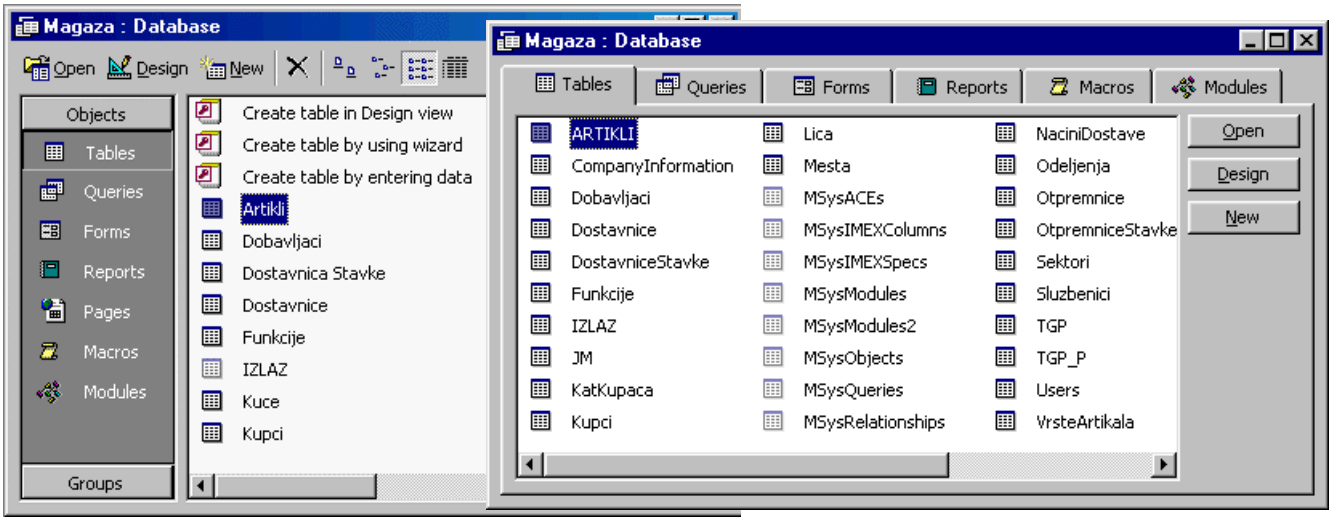
Po otvaranju neke *MS Access* baze podataka na ekranu se pojavljuje *Database Window*. On omogućuje izbor jednog od šest (sedam) tipova objekata aktivne baze podataka, kao i režim (môd) rada u njima:

Open - otvaranje postojećeg,

Design - redizajniranje postojećeg, odnosno,

New - kreiranje novog objekta).

Kod izveštaja opcija *Open* je zamenjena odgovarajućom opcijom *Preview* (kojom se on otvara za pogled), a kod makroa i modula opcijom *Run* (kojom se on pokreće).



Desnim klikom na objekat možete dobiti sve moduse rada sa datim objektom, pa ovaj način treba maksimalno koristiti kod otvaranja objekata za bilo kakvu obradu. Ne treba zanemariti ni ikonice na *Toolbar*-u, a posebno u situaciji kada treba preći iz jednog u drugu modus obrade objekta, a *Database Window* se nije vidljiv. *Toolbar* daje još jednu prednost – preko ikonice možete uvek preći u pretpostavljeni (suprotan) modus (npr. iz *Open* u *Design* režim rada). Na prethodnim slikama je prikazan *Database Window* u *MS Access* 2000 i *MS Access* 97.

Sistem menija

File meni, osim standardnih stavki kao kod svih *Windows* aplikacija, sadrži i stavke:

- *Get External Data* omogućuje uvoz *(Import)* ili povezivanje *(Link Tables)* tabela iz druge *MS Access* baze ili nekog drugog kompatibilnog izvora podataka. Koristeći isti dijalog može se uvesti i bilo koji objekat iz druge *MS Access* baze. *Link*-om povezana *MS Access* tabela ne može se otvarati u dizajn modu (u tom môdu joj se može pristupiti samo otvaranjem matične baze (one u kojoj se tabela nalazi).
- *Save As/ Export* omogućava čuvanje nekog od *MS Access* objekata, pod drugim imenom u tekućoj ili drugoj bazi, kao i u obliku tabele u formatu druge aplikacije.
- *Database properties* je spisak svojstava *MS Access* baze podataka. Može da sadrži informacije o autoru, nazivu, predmetu, ključnim rečima aplikacije, statistike i spisak objekata u bazi, kao i mogućnost kreiranja korisnički definisanih *(Custom)* svojstava baze.

Edit meni sadrži standardne stavke za pokretanje komandi ažuriranja (*Cut, Copy, Paste, Undo, Delete, Rename*) za rad sa objektima iz baze, kao i mogućnost kreiranja prečice (*Shortcut*-a) na dati objekat. Za sve ove alate na raspolaganju su i odgovarajuće stavke *PopUp* menija koji dobijate desnim klikom na objekat, pa ih koristite za brži i kvalitetniji rad, jer je *MS Access*, kao i *OS Windows* predstavlja objektno orijentisano radno okruženje.

View meni sadrži stavke izbora tipa objekta u *database* prozoru, određivanje tipa ikona i njihov raspored, kao i pregled svojstava (*Properties*) objekta i programskog koda objekta (forme i moduli). Preko stavke *Toolbars* omogućen je pristup korisničkom podešavanju sistema menija i paleta alata.

Insert meni sadrži opcije za kreiranje novog objekta u bazi, kao i automatsko generisanje forme ili izveštaja na osnovu selektovane tabele ili upita (kao izvora podataka).

Tools je daleko najznačajniji meni, a u svom prvom delu sadrži poznate *Microsoft Office* alate *Spelling* i *AutoCorrect*. *Spelling* izvršava pravopisnu proveru teksta (za srpski je potrebno instalirati poseban dodatak), dok *AutoCorrect* automatski ispravlja neke karakteristične greške (po želji). Korišćenje *AutoCorrect*-a je slično na nivou celog *Microsoft Office*-a, pa ga ovde nećemo posebno naglašavati. *Database Utilities* sekcija ovog menija, nudi mogućnost konvertovanja (*Convert*) fajla iz ranijih verzija *MS Access*-a u fajl novog formata, a (u verziji MS Access 2000) baza se može konvertovati i u fajl starijeg formata (za *MS Access* 97), mada ne funkcioniše korektno u slučaju aplikacije, ipak, predstavlja alat od izuzetnog značaja. *Compact* obezbeđuje sažimanje baze (kao i opravke fajla posle neregularnih izlazaka iz aplikacije - *Repair*). Sažimanje se može uporediti sa defragmentacijom diska s tim da se odnosi na *MS Access*-fajl. Radi se o sledećem: u toku rada na dizajnu baze, kao i u slučajevima većih zahvata na brisanju i dodavanju slogova, obrisani objekti izazivaju fragmentaciju fajla (samim tim on nepotrebno zauzima više prostora na disku), te ga sažimanje svodi na optimalnu veličinu. Ovo se preporučuje kod velikih izmena u fajlu. *Repair* se automatski startuje ako *MS Access* otkrije da nešto nije u redu prilikom otvaranja baze, dok je u verziji *MS Access* 2000 ovaj alat integrisan uz *Compact*. Ako se baza ponaša nepredvidljivo, ovaj alat se uvek može startovati i manuelno iz menija, a daje dobre rezultate samo u slučaju dobrog relacionog modela (sa što manje propusta u optimizaciji) i uključenog referencijalnog integriteta u što većem broju relacija.

Window meni sadrži stavke za aranžiranje prozora na ekranu (objekti koji se obrađuju).

Help meni ima uobičajene alate za pomoć.

4 TABELA I NJENA ORGANIZACIJA

Tabela, polje, zapis

Da bi obrada informacija u okviru neke aplikacije (u našem slučaju *MS Access*) bila moguća, moramo je prethodno raščlaniti na podatke, koje ćemo zatim smestiti u neku tabelu koja sadrži redove i kolone.

Tabela sadrži podatke o određenom subjektu. Na primer, baza podataka može da sadrži jednu tabelu sa podacima o kupcima (njihova imena, adrese, telefone...), drugu tabelu za proizvode i neku treću za porudžbine.

Polje predstavlja jedinični podatak, kao što je ime osobe, ili naziv ulice, poštanski broj ili naziv proizvoda. Polje se obično pojavljuje kao jedna kolona u tabeli.

Zapis sačinjavaju svi podaci pojedinačne stavke tabele, kao što su svi podaci o jednom kupcu iz tabele kupaca, ili o jednom proizvodu iz tabele proizvoda. Zapis se, u vizuelno predstavlja kao red. Zapisi se često nazivaju slogovima ili rekordima (*Records*) ili n-torkama, gde je n-broj polja u slogu.

Atribut, domen i relacija

Atribut element informacije kojim je jednoznačno uređena vrsta svojstva.

Domen skup svih vrsta vrednosti koje određeni atribut može primiti.

Relacija skup uređenih n-torki.
Relacija ima sledeća svojstva:

- šema relacije ne sadrži dva jednaka naziva atributa,
- redosled kolona u relaciji nije bitan,
- relacija ne sadrži dve jednake n-torke,
- redosled n-torki u relaciji nije bitan.

Stepen relacije broj uređenih n-torki.

Relaciona šema konačan skup naziva atributa zajedno sa skupom ograničenja.

Kreiranje tabela Wizard-om

MS Access, pored mogućnosti kreiranja baze podataka *Wizard*-om (čarobnjak), pruža na raspolaganje i *Wizard* za kreiranje tipskih tabela podataka. U fazi savladavanja i istraživanja u oblasti projektovanja baza podataka sigurno je da će korišćenje ovog alata doprineti bržem uočavanju mogućnosti *MS Access*, ali se u profesionalnom radu nameće potreba za kreiranjem sopstvenih tipskih tabela.

To znači da ćete se u svom daljem radu u *MS Access*-u sve više oslanjati na sopstvene snage, a manje na automatizme, bar kod upravljanja tabelama i upitima. Čarobnjaci kod izrade formi i izveštaja su nezaobilazni na svim nivoima poznavanja *MS Access*.

Indexiranje i ključevi

Indexiranjem se omogućava da se, u daljem korišćenju tabele kao izvora podataka, može brže sortirati, pretraživati i obrađivati u svakom drugom obliku – po polju podataka koje je indeksirano. Indeksiranje se preporučuje za polja numeričkih podataka i to samo u situacijama kada se nad njima planira neki oblik pretraživanja. Uključivanje indeksiranja po polju podatka može dovesti do znatnog uvećanja fajla baze podataka, a dešava se obično kod tabela koje imaju veliki broj slogova podataka (u desetinama hiljada i više).

Isključivanje indeksiranja može prouzrokovati spor odziv upita kod kojih je primenjen neki od oblika sortiranja i kaskadnog grupisanja, pa je na programeru da odabere pravi način u konkretnom slučaju. Kada je po datom polju uključen primarni ključ – update podataka je nešto sporiji, ali se zato svaki upit po ovom polju drastično ubrzava.

Ovde od presudnog značaja može biti iskustvo kolega, pa ih, u svakom slučaju, saslušajte.

Kardinalnost - broj n-torki ili slogova (predstavljeni po kolonama) jedne relacije (ili tabele).

Polje primarnog ključa - Ovo polje jednoznačno identifikuje svaki zapis u tabeli. Na osnovu sadržaja ovog polja može se tačno odrediti o kom zapisu se radi. Iz toga sledi da je primarni ključ, u stvari, atribut neke tabele (relacije) koji identifikuje jedan slog (n-torku) i naziva se još ključem relacije.

Prost ključ - kada u jednoj tabeli postoji polje koje u potpunosti može odrediti jedinstvenost sloga, onda se takav ključ naziva prostim.

Entitet RADNIK sadrži atribute RadnikID, Prezime, Ime, Zanimanje i MB.

RADNICI

RadnikID	***Prezime***	***Ime***	***Zanimanje***	***MB***
001	Petrović	Petar	novinar	1011963325641
002	Stojanović	Aleksandar	elektrotehničar	0209948025632
003	Slavković	Svetlana	prevodilac	2612975361258
004	Topolac	Životije	profesor	1809958521438
005	Stojanović	Aleksandar	geodeta	1111197211121

Ovde se vidi kako polje primarnog ključa (*RadnikID*) jednoznačno određuje slog; zamislimo da je kolona zanimanje u nekom upitu izostavljena. U tom slučaju, na bazi vrednosti polja *RadnikID* možemo zaključiti o kom Stojanović Aleksandru se radi, tj. koje je njegovo zanimanje, odnosno da li je u pitanju Stojanović Aleksandar- elektrotehničar ili neko drugi sa istim imenom i prezimenom ko je po zanimanju geodeta.

Složen ključ - Kada se jedinstvenost sloga ne može u potpunosti odrediti preko jednog polja, tada se mora koristiti složen primarni ključ. Ako u tabeli RADNICI imamo samo polja Prezime i Ime, tada bismo morali uvesti i polje SrednjeSlovo jer, kao što vidimo Prezime i Ime ne mogu u potpunosti odrediti jedinstvenost sloga (jer se mogu pojaviti dva radnika sa istim imenom i prezimenom). Tada primarni ključ čine zajedno Prezime, Ime i SrednjeSlovo, što predstavlja složeni primerni ključ, jer ga čini više polja.

Na primer, ako relaciji RADNICI dodamo atribut MB (matični broj), tada je i ovaj atribut kandidat za ključ, odnosno pomoću njega se takođe može odrediti jedinstvenost n-torke. Obično se, u ovakvim situacijama, jedan atribut bira za *primarni ključ*, a ostali neizabrani kandidati se nazivaju *alternativnim ključevima*.

Spoljni ključ - Pored primarnog ključa, relacija može imati i spoljni ključ. O čemu se zapravo radi?! Ponekad se dešava da jedna relacija ne može egzistirati bez druge, na primer stavka fakture ne može da postoji bez fakture. Recimo da relacija FAKTURE ima ovaj oblik:

Fakture (BrojFakture, Datum, Partner).

BrojFakture je primarni ključ jer određuje jedinstvenost sloga. Faktura ima svoje stavke, pa iz toga proizilazi da relacija FaktureStavke ima ovakav oblik:

FaktureStavke (BrojFakture, BrojStavke, Artikal, JedinicaMere, Količina).

Kako stavka bilo koje fakture ne može da postoji bez fakture kojoj pripada, tako ni relacija *FaktureStavke* ne može da egzistira bez relacije *Fakture*. Ove dve relacije povezane su preko ključa *BrojFakture* koji je zajednički za obe relacije. Na osnovu njega se zna koja stavka pripada kojoj fakturi. Zato se kaže da, u ovom slučaju relacija *FaktureStavke* nasleđuje atribut BrojFakture od relacije *Fakture* i on je za relaciju *BrojFakture - spoljni ključ*. Spoljni ključ služi za uspostavljanje relacija između tabela u relacionim bazama podataka.

VEŽBA: Kreirajte tabelu KUPCI(Naziv, Lice, Adresa, Mesto, Telefon). Složen ključ čine polja Naziv i Mesto, jer se u istom mestu (gradu) ne sme pojaviti kupac sa istim nazivom firme,ali se dozvoljava da ista firma postoji sa istim imenom u više mesta.

5 OSNOVE UPRAVLJANJA TABELAMA

Tabela predstavlja kolekciju podataka određene tematike i primarni izvor podataka za bazu. Sastoji se iz slogova (redova, eng. *record*) i polja (*fields*). Ona je osnovni objekat baze podataka. Podaci kojima se fizički manipuliše jedino postoje u tabelama, dok su ostali objekti baze samo način da se podaci iz baze, izabrani po nekom kriterijumu, sortirani i grupisani (po potrebi), prikažu na određeni način. Zato je kreiranje i definisanje karakteristika tabela, kao i uspostavljanje relacija između njih, jedan od najvažnijih zadataka prilikom projektovanja baze podataka.

Podaci o Firmi : Table

ID_Fi	Maticni I	Naziv Firme	Adresa	ID_Mesto	Telefon
2168	601233	POZORIŠTE NA TERAZIJAMA	TERAZIJE 29	BEOGRAD	011/334-037,338-907
2169	601101	POŠTANSKA ŠTEDIONICA	27. MARTA 71	BEOGRAD	011/3248-111 3248-610 32
2170	601104	PPT INŽENJERING ,D.D.	BULEVAR VOJVODE MIŠIĆA 37-3	BEOGRAD	011/235-1650,235-1890
2171	600801	PRAG, HOTELSKO UGOSTITELJSKO I TUR. PRE.	NARODNOG FRONTA 27	BEOGRAD	011/687-355
2172	601231	PRAKTIČNA ŽENA, NIP	NEMANJINA 4	BEOGRAD	011/641-155/631-653
2173	600118	PRAM, PP	ALEKSE NENADOVIĆA 32	BEOGRAD	011/3220-188, 459-406
2174	601201	PRAVNI FAKULTET UNIVERZITETA U BEOGRADU	BULEVAR REVOLUCIJE 67	BEOGRAD	011/341-501
2175	601103	PRED. GRADSKIH PIJACA I TRŽNIH CENTARA	ŽIVKA KARABIBEROVIĆA 3	BEOGRAD	011/422-566
2176	601104	PREDNAPREGNUTI BETON, PRED.ZA PROJEKTOV.	UČITELJSKA 60/I	BEOGRAD	011/4886-810, 4887-004
2177	601103	PREDUZEĆE PLAVIJANKA PLAV	VELIZARA KOSANOVIĆA 2	BEOGRAD	011/413-858 I 410-054
2178	601109	PRENAD,D.O.O	DOSITEJEVA 1/III	BEOGRAD	011/626-569,181-108
2179	600713	PRESEDAN M, PP	MIŠKA KRANJCA 7	BEOGRAD	011/583-346, 582-493
2180	601231	PRESS EXPRES (DIREKTOR MITAR DANILOVIĆ)	ŠUMADIJSKI TRG 6A PF2024	BEOGRAD	011/556-598,559-845
2181	600605	PREVOZ, PLJEVLJA, PREDSTAVNIŠTVO	BALKANSKA 48	BEOGRAD	011/687-164
2182	600134	PRIMA GRAFIKA	SARAJEVSKA 1	BEOGRAD	011/642-666,339-422
2183	600501	PRIMA KOMERC	TADEUŠA KOŠĆUŠKOG 70	BEOGRAD	011/187-333, 187-496, 637
2184	600716	PRIMA MEDICAL, CO	CARA DUŠANA 72	BEOGRAD	011/639-082, 187-496, 637
2185	600902	PRIMA, OMLADINSKA ZADRUGA	VLAJKOVIĆEVA 9	BEOGRAD	011/347-516
2186	600720	PRIMAR EXPORT-IMPORT D.O.O.	CARA LAZARA 5	BEOGRAD	011/628-653,627-345
2187	600716	PRIMAX, VELEDROGERIJE I APOTEKE	NEMANJINA 2	BEOGRAD	011/644-968
2188	600702	PRIMERA	TEODORA MIRIJEVSKOG 64	BEOGRAD	011/444-7834,457-542,411
2189	601231	PRINCIP, IP	KOLARČEVA 7/6	BEOGRAD	011/3248-496
2190	601102	PRIREDIVANJE IGARA NA SREĆU DP	SVETOZARA MARKOVIĆA 40	BEOGRAD	CENTR. 011/334-876

Record: 3 of 26431

Nije neophodno da svi podaci, koje planirate da vodite u bazi podataka, budu u jednoj tabeli. Obradom postojećih uvek se mogu sračunati potrebni podaci (i dobiti informacije), a razvrstavanjem po tabelama možete postići veće efekte u funkcionalnosti i lakom održavanju tabela, a samim tim i baze podataka.

Kreiranje tabela

Početni korak za kreiranje tabele je komanda *New* u *database* prozoru (kada je aktivna kartica *Tables*). Ovim se otvara dijalog koji nudi pet načina u kreiranju tabele i to:

Datasheet View omogućava direktno unošenje podataka u tabelu, pri čemu program sam određuje tip polja. Ime polja se daje opcijom *Rename Column* iz menija koji se dobija desnim klikom na naslov kolone. Ovaj način se koristi u dva slučaja i to: a) kada još nije poznata struktura tabele i b) kada je potrebno ubaciti kopirane podatke iz nekog drugog programa (npr. *MS Excel*). *MS Access* će svim poljima dati imena kao Field1, Field2,..., a na osnovu sadržaja unetih po poljima, pokušati da odredi neka svojstva polja, kao što je maksimalna dužina i format podatka. Ovde se podrazumeva da su sva polja tekstualnog formata i opciono unete dužine u samom *MS Access*-u. Ovaj pristup u kreiranju tabele treba izbegavati, jer se njime skoro ništa ne definiše.

Design View vodi u standardno kreiranje tabele detaljnim izborom svojstava polja podataka, kao što su imena polja, tip podataka i veličina polja, određivanje primarnog ključa, indeksiranje, graničnih uslova za unos podataka i slično. Poznavanje ovog pristupa je najvećeg značaja, a ujedno je to najkompleksniji, najčešći i najdetaljniji pristup.

Table Wizard prepušta formiranje tabela *Wizard*-u, koji to čini pozivajući se na biblioteku već formiranih tabela. Ova mogućnost je korisna kada želimo da kreiramo neke manje-više standardne objekte kao što su adresari, podsetnici, računi, izveštaji, fakture, evidencije troškova i sl. Ovaj pristup je koristan u učenju i savladavanju "tajni" *MS Access*-a, ali je njegov nedostatak u tome što nudi mnogo toga, a vi najčešće ne možete da pronađete baš ono što vam treba.

Import Table daje mogućnost uvoza tabele iz nekog drugog izvora podataka koji je kreiran prema nekom od standarda za datoteke i baze podataka. Može biti reč o *MS Access* bazi, ali i mnogo drugih organizovanih tabela, kreiranih programima kao što su: *MS Excel*, *dBASE*, *Cliper*, *FoxPro*,..., kao i fajlove tekstualnog formata, gde je za podelu na kolone (polja) potrebno definisati širinu svake kolone pojedinačno. Treba imati na umu da se na ovaj način stvara kopija importovane

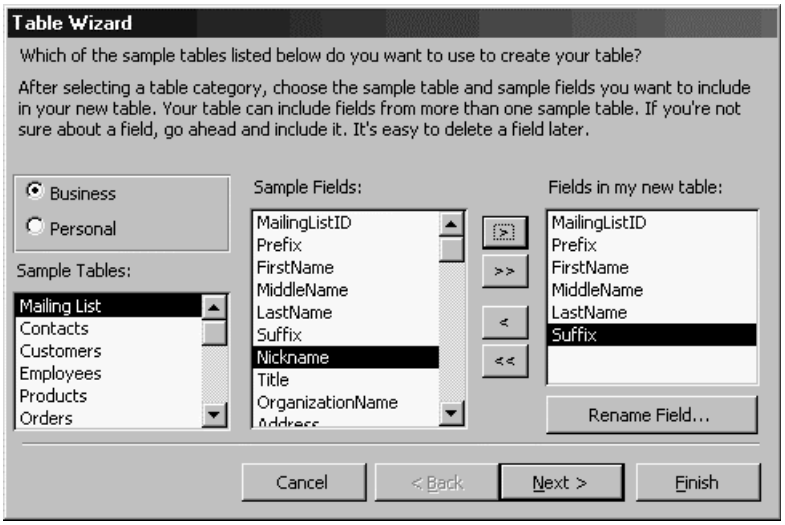
tabele, pa se u slučaju rada u lokalnoj mreži ili IS-u može desiti decentralizacija (rasipanje) podataka, odnosno ponavljanje istih podataka na više mesta, što za održavanje takvog sistema može biti fatalno.

Link Table daje mogućnost povezivanja sa tabelom (obično više njih) iz druge *MS Access* baze (retko kada se koristi neki drugi format baze), pri čemu tabela ostaje u matičnoj bazi, a linkom se omogućuje veza na nju u cilju pregleda i ažuriranja podataka. U ovoj situaciji, pristup dizajnu tabele nije moguć iz baze koja ima link na datu tabelu, već samo iz matične (izvorne) baze. Dizajnu tabele se ne može pristupiti dok se svi aktivni linkovi na odnosnu tabelu ne isključe, što znači da pri dizajnu tabele niko (kroz mrežu) ne može da je ažurira. Kako mrežne aplikacije uglavnom rade na više tabele i više baza, u ovoj situaciji je potrebno uputiti korisnike da privremeno ne otvaraju podatke koji se nalaze u ovoj tabeli (jer će biti upozoreni da to trenutno nije izvodljivo), a posle izvršenih ispravaka u dizajnu – nastavljaju sa uobičajenim radom. Ujedno ovo predstavlja i jedinu situaciju u kojoj, korisnik baze podataka kroz računarsku mrežu, sme da bude ometan u njenom korišćenju. Mnogi će vam priznati da bez *Link*-a nema ni mrežne orijentacije baze podataka i da je to tek kamen temeljac za dobar informacioni sistem.

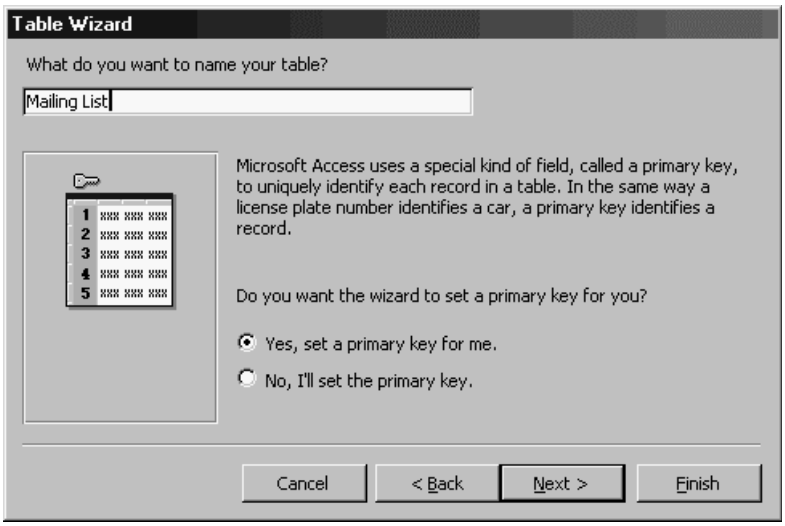
U kompleksnijim bazama podataka se obavezno praktikuje da nisve potrebne tabele ne budu u odnosnoj bazi, jer *MS Access* omogućava njihovo povezivanje (*Link*) kroz baze. Ovo je od ključnog značaja za projektovanje aplikacija za korišćenje u lokalnoj mreži ili informacionom sistemu. Održavanje i razvoj ovakvih sistemaje mnogo lakše za programera, a korisniku daje mogućnost kontinuiranog rada bez obzira na zahvate na administriranju.

Table Wizard

se koristi u fazi učenja, pa bi bilo dobro da ovde prikažemo njegovo korišćenje. Komercijalne (poslovne) baze podataka, koje se eksploatišu i po 24 na dan, moraju se ozbiljnije projektovati, pa prilikom kreiranja tabele u njima nećete moći (a ni želeti) da koristite *Wizard*-e. Tabele i upiti su toliko značajni, objekti baze podataka, da traže više poštovanja, vremena i znanja, od pukog *Wizard*-a, jer se korišćenjem raspoloživih automatizama programer "otuđuje" od svoga dela.

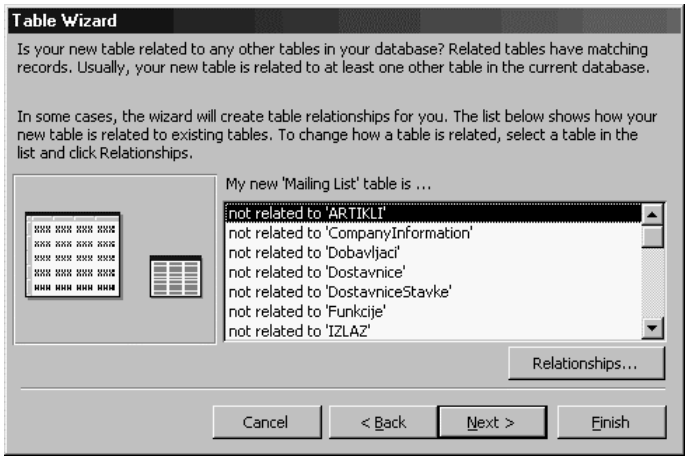


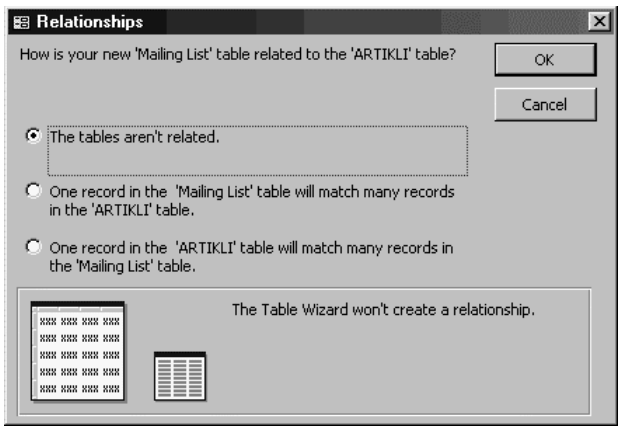
U prvom dijalogu *Table Wizard*-a izaberite tip tabele (poslovna ili personalna), zatim neku od predloženih tabela, polja iz izabrane tabele i ... kreirali ste listu polja vaše nove tabele (poslednja lista), a preko komandnog tastera "Rename Field" možete promeniti ime izabranih polja.



U drugom dijalogu *Table Wizard* vam nudi da unesete ime tabele i (opciono) deklarišete polje primarnog ključa.

U trećem dijalogu *Table Wizard*-a možete pregledati relacije kreirane tabele sa drugim tabelama u bazi, a potom preko tastera "*Relationships..*" (eventualno) možete zadati parametre neke od relacija (kao u prikazanom četvrtom dijalogu).

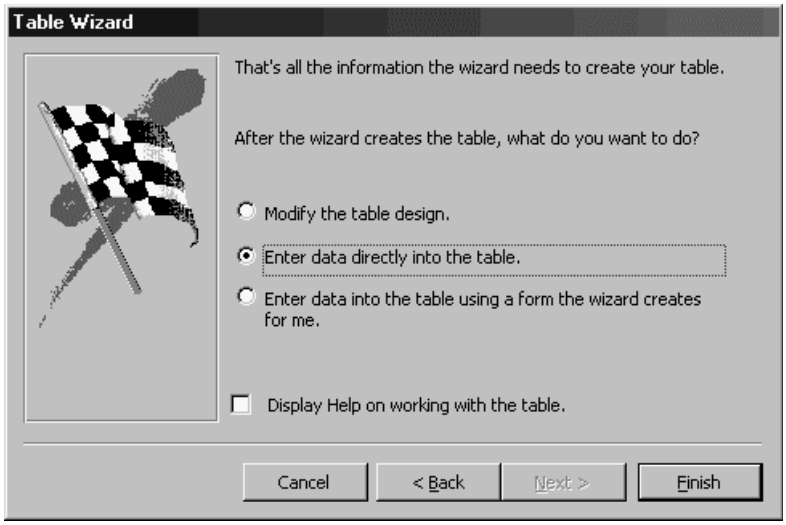




U petom dijalogu se možete odlučiti na koji način ćete otvoriti kreiranu tabelu:

- u *Design View* môdu,
- *Datasheet View* môdu ili
- dozvoliti *MS Access*-u da kreira i formu (obrazac) za unos podataka ove tabele (samo u verziji 2000).

Na sličan način se koriste i svi ostali *Wizard*-i: *Query Wizard*, *Form Wizard*, *Report Wizard*.



Dizajn tabele

Način na koji se najčešće dizajniraju tabele je *Design View*. Tu je omogućen detaljan opis i definicija svakog polja nove tabele, kao i kasnije modifikacije postojeće tabele. U gornjem panelu otvorenog prozora upisuje se:

Field Name ime polja, kojim se definiše ime atributa u tabeli. Iz mnogih razloga (koje ovde nećemo pominjati) poštujte sledeća pravila kod zadavanja imena polja:

a) koristite što kraće ime uz kombinovanje velikih i malih slova,
b) ne koristite specijalne znake osim "_",
c) ne koristite YU slova.

Iako, i bez ovih pravila, sve može da funkcioniše u lokalnom okruženju (vaš *MS Access* i vaš PC i vaša lokalna mreža) ipak kod distribucije (publikovanja) baze podataka ili njenog povezivanja sa nekom aplikacijom u razvoju (npr. pod *VisualBasic*-om ili operativni sistem UNIX) možete imati gadnih problema. Ista pravila treba koristiti i kod imenovanja fajla baze i svih objekata u bazi. Ovo ne treba mešati sa unosom podataka i komentara, gde se može koristitibilo koje pismo ili jezik (omogućen operativnim sistemom).

Data Type tip podatka, koji se čuva u tom polju je najvažnije svojstvo polja, pa mu posvetite veliku pažnju. Ovde je važno što više izbegavati *Text* tip, jer se njegovim korišćenjem najmanje štedi prostor, a baza gubi na performansama (zauzeće, brzina rada, korisnička orijentacija,...). Ponekada ni sam korisnik to neće primetiti dok ne vidi i drugu situaciju. Primer za ovo može biti poštanski broj mesta, koji treba obavezno definisati kao broj (*Number*), a ne kao tekst (*Text*).

Description tekstualni opis podatka dužine do 255 karaktera, a njegovo upisivanje se savetuje, jer se on, u toku rada korisnika na datom polju podatka, ispisuje u statusnoj liniju i korisniku pruža dodatne informacije o tom polju.

MS Access podržava sledeće tipove podataka (*Data Type*), a u dodatku su dati i opsezi za numeričke tipove:

Text tekst (slova, brojevi i svi specijalni znaci) sa kojim se ne vrše računske operacije, a dužine do 255 karaktera.

Memo dugačak tekst (slova, brojevi i svi specijalni znaci) dužine do 65535 karaktera. Ovakav tip tekstualnog podatka ne može ući u neko sortiranje, pretraživanje ili grupisanje, što treba imati u vidu.

Number brojevi koji se upotrebljavaju u aritmetičkim izrazima, a veličina polja (*Field Size*) zavisi od "podtipa" koji može biti: *Byte, Integer, Long Integer, Single, Double, Replication ID, Date/Time, Currency, Auto Number* (koji može biti *Long Integer* ili *Replication ID* opsega) ili *Yes/No*.

OLE Object Objekat kao *MS Word* dokument, *MS Excel* tabela, zvuk, grafikon, slika... povezan je (*Linked*) ili ugrađen (*embedded*) u *MS Access* tabelu.

Lookup Wizard daje mogućnost da se kreiraju polja koja omogućavaju izbor vrednosti iz druge tabele ili iz (ograničene) liste vrednosti upotrebom grafičkih kontrola *ComboBox* ili *ListBox*. Izborom ove stavke startuje se *Lookup Wizard*, kojim se definiše vrsta izvora, način popunjavanja i broj polja. Zbog značaja i *Lookup Wizard*-a o njemu će biti reči u nastavku.

U zavisnosti od izbora tipa podatka u donjem panelu *Design View* prozora prikazuje se dodatna lista svojstava svrstanih u dve grupe (kartice): *General* i *Lookup*.

GENERAL kartica sadrži listu opštih svojstava polja:

Field Size je veličina polja; podrazumevani tip i dužina se podešava opcijama u *MS Access*-u. Najčešće je to 50 karaktera za tekst, a, *Long Integer* za brojeve.

Decimal Places je svojstvo specifično za numeričke podatke i određuje broj cifara iza decimalne tačke.

Format definiše poseban izgled tekstualnog polja, dok je za brojeve predloženo nekoliko standardnih tipova *(General Number, Currency, Fixed, Standard, Percent, Scientific),* kao i za *Date/Time (General*, Long*, Medium*, Short*).* Ovi formati u velikoj meri zavise od podešavanja u *Regional Settings* servisu *Control Panel*-a, pa je pravi trenutak da u tom servisu sve podesite na formate ispisa podataka u našem regionu i to:

1. *Number*: sa decimalnim zarezom i tačkom kao separatora za grupe, a tačka-zarez za separator liste
2. *Currency*: simbol za valutu podesite na "din", a ostalo kao i za *Number*
3. *Time*: "HH:mm:ss" i za separator dvotačku
4. *Date*: "dd.mm.yyyy", bez tačke na kraju i tačku za separator

Input Mask definiše masku za unos podataka, koju korisnik mora ispoštovati prilikom unosa podatka, a korisna je u slučajevima unosa: broja telefona, broja pošte, matičnog broja i sl.

Caption po potrebi definiše naslov polja. Ovaj tekst će biti ubačen pri kreiranju grafičko-upravljačke kontrole u nekoj formi koja prikazuje podatke iz ove tabele. Ukoliko se ovde ne unese ništa, uzima se već postavljeno ime polja u *Field Name*, pa se njegov unos savetuje u slučajevima kada, a ako ne bude uneto podrazumeva se da ima vrednost kao i ime polja (*Field Name*).

Default Value predstavlja vrednost koja će se automatski upisati u polje pri popunjavanju novog zapisa.

Validation Rule je izraz koji definiše zahteve koje mora da ispuni uneti podatak. Obično su uslovi vezani za brojevnu vrednost i opseg kome dati podatak pripada, a može se rešiti i pitanje da li je podatak uopšte unet, jer ako nije unet ima problematičnu "vrednost" *Null*.

Validation Text je tekst poruke koja se pojavljuje ukoliko uslov *Validation Rule* nije zadovoljen. Kreiranjem potpune validacije izbegavaju se komentari, koje korisnik dobija na engleskom.

Required određuje da li je u tom polju unos podatka obavezan (*Yes*) ili ne (*No*).

Allow Zero Length je svojstvo specifično za tekstualne podatke i određuje da li je dozvoljeno postojanje stringa nulte dužine ("") u polju.

Indexed je svojstvo kojim se uključuje indeksiranje na određeno polje u cilju izbegavanja dupliciranja podataka i bržeg pretraživanja ili sortiranja po specificiranom polju. Efekat indeksiranja se može sagledati tek kod rada "velikih" tabela (par hiljada slogova) i baza podataka sa više ovakvih tabela, a ponuđene su tri opcije:

1) polje nije indeksirano (*No*),
2) polje je indeksirano, a duplikati vrednosti su dozvoljeni (*Yes-Duplicates OK*) ili
3) polje indeksirano, ali duplikati nisu dozvoljeni (*Yes-No Duplicates*).
   Kod polja koje predstavlja primarni ključ mora biti odabrano (*Yes-No Duplicates*).

Unicode Compresion omogućava smanjenje prostora koji će biti zauzet *Unicode Text*-om. Ovo svojstvo je omogućeno u verziji *MS Access* 2000, a iz razloga što *Unicode Text* zauzima dvostruko veći memorijski prostor (1 znak zauzima 16, a ne 8 bita kao *ASCII Text*).

LOOKUP kartica sadrži svojstva u slučaju da se podataka u tekućem polju bira iz liste vrednosti. Sve započinje izborom tipa kontrole koja prikazuje listu raspoloživih vrednosti, tako da u zavisnosti od vrste podataka koje polječuva može biti *Text Box* (nije *Lookup* polje), *ComboBox*, i *List Box* (standardno), kao i *CheckBox* (za *Yes/No* tip podataka). *MS Access* omogućava da se ovde detaljnije odredi izvor podataka za odnosno polje, mada je *Lookup Wizard,* ipak, za početak jednostavniji.

Lookup Wizard

predstavlja skoro idealan način da inicirate prvu relaciju povezujući dva polja između dve tabele u vašoj bazi, ali obavezno ispoštujte sledeću proceduru:

1) kreirajte izvornu tabelu koja obavezno treba da sadrži primarni ključ sa *AutoNumber* ili *Number* tipom podatka i bar još jedno polje (obično tekstualno i male dužine);
2) u izvornu tabelu Zanimanja(ZanimanjaID, Zanimanje) unesite nekoliko tekstualnih podataka (npr. "pravnik", "ekonomista", "novinar", "profesor" i "pekar";
3) kreirajte novu tabelu Radnici(Prezime, Ime, Zanimanje), pa za jedno polje podatka (Zanimanje) iskoristite *Lookup Wizard* i zadajte za izvor podataka liste – prethodno kreiranu tabelu (oba polja);
4) ako je potrebno zadajte drugo ime za polje koje ste kreirali, a zatim sačuvajte tabelu (dizajn);
5) otvorite istu tabelu za ažuriranje, pa unesite nekoliko slogova podataka.

Uočite da će se *AutoNumber* (ujedno i ključ) sam sakriti, pa ćete u korišćenju polja sa vezom na drugu tabelu videti samo tekstualni prikaz, a u polje podatka će biti upisana numerička vrednost odgovarajuće stavke koju odaberete iz liste. Pri tom, tabelu izvora podataka za listu možete ažurirati, što će se odražavati i na stavke u listi. U slučaju da vam ovo ne uspe, preostaje da sami podesite parametre u kartici LOOKUP.

Imenovanje objekata

Imena objekata (fajl baze podataka, objekti u bazi i polja u tabelama) je najbolje zadavati primenjujući sledeća pravila:

- kombinovati VelikaMala slova bez odvajanja na više reči i bez specijalnih znakova
- VELIKIM slovima imenovati one važnije objekte (mada suštinske razlike nema, ali se brže uočavaju u slučaju većeg broja objekata);
- primeniti imenovanje oblika '_IME_' - kada objekat treba postaviti kao prvi u nizu (specijalni objekat od značaja za projektovanje baze ili aplikacije)
- primeniti imenovanje oblika 'Ime_Sub' - kada odnosni objekat (tabela, forma) predstavlja podobjekat
- objekte, planirane za uklanjanje (brisanje), imenovati sa prefiksom u obliku 'zzIME', jer će se naći na kraju liste objekata (ako je izabran uobičajen prikaz objekata po imenu - *By Name*).
- izbeći kreiranje tabele i upita sa istim imenom, jer može doći do greške prilikom pozivanja (*MS Access* i tabelu i upit smatra izvorom podataka jednakog prioriteta), pa se praktikuje imenovanje upita uz prefiks 'qry' (kao npr. 'qryArtikli'). Mada se prefiksi koriste i kod imenovanja svih drugih objekata (tab, frm, rpt, mod i sl. čno), kod imenovanja upita to nije od samo formalnog već od suštinskog značaja za korektan rad baze.

Sigurnost baze i brisanje objekata

Kada započinjete novi projekat (sav rad na novoj bazi ili aplikaciji) najpre se odlučite za kreiranje novog foldera (u okviru vaše zaštićene lokacije), a zatim u njemu kreirajte uobičajene foldere (kao npr: DOCs, Backup, Old, Trash,...).

MS Access baza podataka predstavlja jedan fajla, pa se lako može desiti da je, posle višemesečnog rada, i obrišete kao običan fajl. Posebno treba da izbegavate brisanje bez prebacivanja u *Recycle Bin* vašeg *Windows*-a, bilo preko <Shift>+<Delete> ili podešavanjem *Recycle Bin* na ovu nesretnu opciju, pa ako bazu i obrišete- znate da je u korpi. Interesantna je i činjenica da se fajl baze podataka mnogo lakše briše (preko *Windows Explorer*-a) od mnogih objekata same baze (preko *MS Access*-a), a da bi obezbedili bazu od brisanja iz *Windows Explorer*-a (koje bi mogao da obavi slučajni korisnik vašeg računara) nabavite neki od programa za zaštitu pristupa folderima ili iskoristite mogućnosti operativnog sistema u oblasti personalizacije (Passwords, Users, SaveSource i slično).

Pošto, objekti obrisani iz baze, ne mogu biti vraćeni, to nameće potrebu da se, prilikom uklanjanja nepotrebnog objekta, a u cilju sigurnijeg rada, primene neka pravila. Ono što je najvažnije- obratite pažnju na svaku poruku kojom vas *MS Access* upozorava da će objekat biti izbrisan.

Jedno od zlatnih pravila je da objektu dodelite neko ime koje će vas asocirati da je taj objekat planiran za brisanje, ali ga ne treba brisati dok god se ne uverite da sve u modelu baze funkcioniše korektno. Prilikom promene imena objekta treba voditi računa i o objektima sa kojima je preimenovani objekat u vezi, kao naprimer: tabele u relaciji, ugnježdeni upiti, podobrasci (*Subform*/*Subreport*).

Drugi način je da, sve nepotrebne objekte prebacite u drugu bazu (*Copy*/*Paste*), pa ako vam neki od starih objekata (ili neki njegov deo) zatreba - znate da ga možete naći na sigurnom mestu. Ovo je navika mnogih profesionalaca, a ono o čemu treba da vodite računa je da svaki od projekata ima jednu ovakvu bazu. Važno je držati se i pravila kod imenovanja ovakvih (*Recycle*) baza, da ne dođete u situaciju da ne znate koja od baza je "ona prava".

Čak je i redosled brisanja objekata nekada od važnosti, pa se bezopasnim smatra brisanje sloga podataka (korisnički pristup). Ovo brisanje se može povratiti (preko *Paste*) ako je za brisanje korišćena akcija *Cut*, ali se ni u tom slučaju ne garantuje istovetnost brisanih slogova (slučaj polja tipa *AutoNumber*).

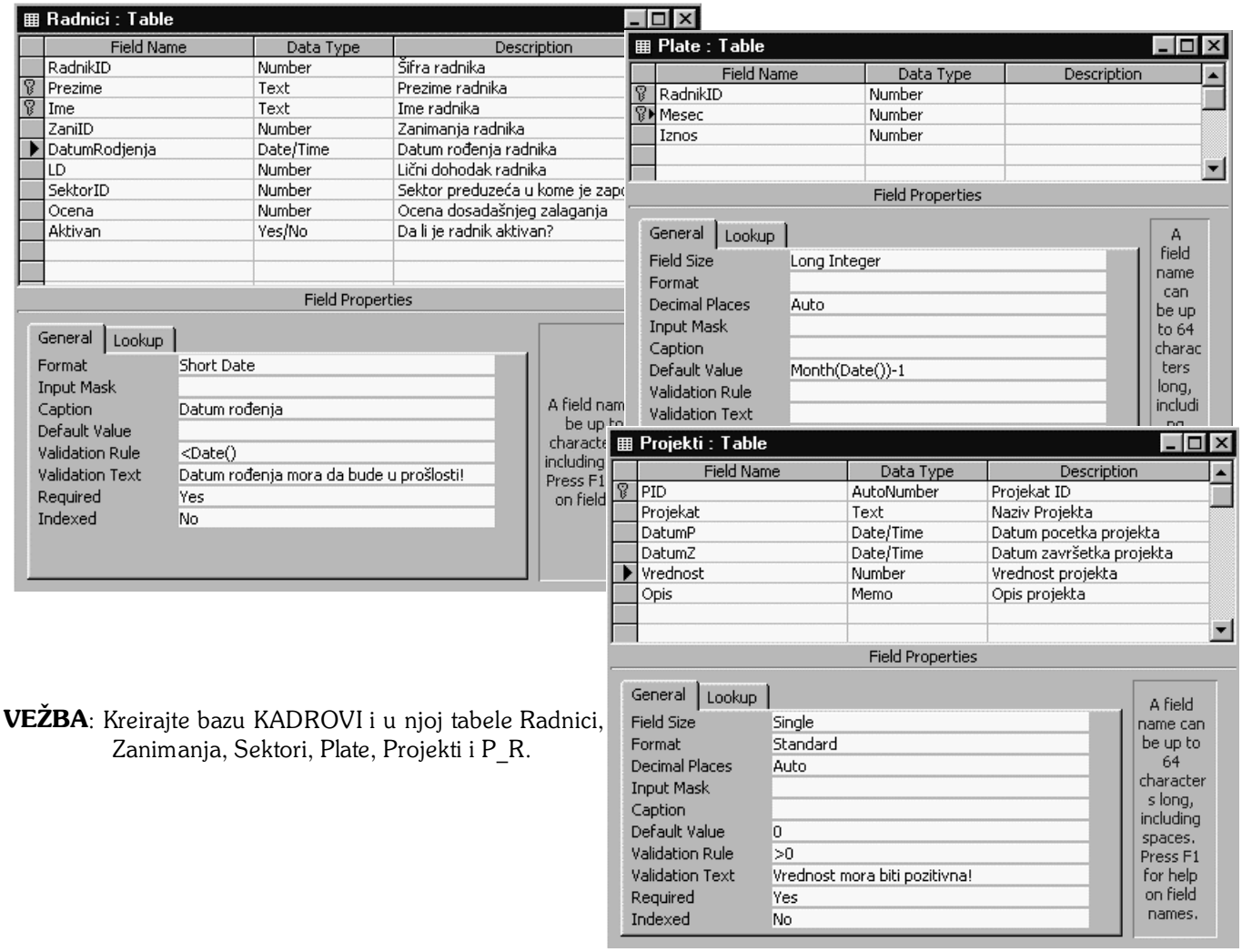
Ostala brisanja su u nadležnosti administratora baze, gde treba voditi računa da se brisanjem objekata brišu i svi podaci koje oni čuvaju, a brisanje obavljati redom:

1. polje u tabeli (*Design View*)
2. relacija između dve tabele (*Relationships*)
3. polje primarnog ključa (*Design View*)
4. tabela (*Database Window*)

Brisanje formi i izveštaja nije ni izdaleka tako opasno, jer se za njihovo kreiranje mogu koristiti čarobnjaci, a koji zahtevaju postojanje izvora podataka (tabele i upiti). Brisanje upita ili modula (procedura kao njihovih delova) može predstavljati problem za one koji nemaju iskustva u njihovom kreiranju, a ono što je u ovom slučaju važno je da model baze (tabele i relacioni model) ne može biti narušen, pa se brisanje smatra bezopasnim.

SAVET: Polja podataka, koja se mogu dobiti obradom (izračunavanjem) podataka iz postojećih polja - ne treba kreirati. Tako npr. u tabeli Stavka(Artikal, Komada, Cena) ne treba kreirati polje Vrednost, jer se ono može uvek dobiti proizvodom polja Komada i Cena. Izračunata polja dovode do problema redudanse i suvišnog naknadnog ažuriranja.

Na sledećim slikama dat je primer dizajniranja tabela, gde se mogu prepoznati značenja atributa i izgledi tabela u dizajn modu.



VEŽBA: Kreirajte bazu KADROVI i u njoj tabele Radnici, Zanimanja, Sektori, Plate, Projekti i P_R.

6 NORMALIZACIJA BAZE PODATAKA

Ovo je proces podešavanja strukture baze podataka kako bi njene performanse bile poboljšane, kako u pristupu podacima, tako i u pogledu integriteta i brzine procesiranja (sreće se i pod nazivom: kreiranje relacija normalizacijom). Sva pravila normalizacije podataka u nekoj bazi ne moraju se obavezno primeniti, ali u kompleksnijim i profesionalnim bazama podataka teško ih je izbeći.

Svako od navedenih pravila podrazumeva pravila nižeg reda. Ova pravila bi trebalo da prepoznate u kreiranju baze (Kadrovi).

Pravilo 1: Eliminisanje grupa podataka koje se ponavljaju

Prvo pravilo normalizacije baze podataka sastoji se u tome, da morate napraviti odvojenu tabelu za svaki set kolona koje su u relaciji pa svakoj tabeli date primarni ključ. Za baze podataka koje poštuju ovo pravilo se kaže da su u prvoj normalnoj formi.

Dakle, ako u tabeli imate niz polja (sličnog imena) uklonite ovu ponavljajuću grupu tako što ćete kreirati novu tabelu za ponavljajuće podatke i povežite je relacijom sa ključnim poljem iz prve tabele.

PRIMER: Ako tabelu PLATE(RadnikID, Ime, Mesec, Sektor, Rukovodilac, LD) svedemo na PLATE(RadnikID, Mesec, Sektor, LD) izbećićemo ponavljanje grupa podataka RadnikID i Ime, kao i Sektor i Rukovodilac.

Pravilo 2: Eliminacija redudantnih podataka

Redudantnim podacima se smatraju oni podaci koji su nepotrebno smešteni na više mesta u bazi podataka. Drugo pravilo normalizacije baza podataka kaže da, ako polje samo delimično zavisi od više-vrednosnog ključa, premestite ga u drugu (posebnu) tabelu.

Dakle, nemojte smeštati iste podatke na dve različite lokacije (tabele). Ovo može voditi u greške ažuriranja ili brisanja. Ako su ekvivalentni elementi podataka uneseni u dva polja, uklonite drugi element podataka, kreirajte novu tabelu sa odnosnim elementom i pratećim ključnim poljem, a zatim smestite ključno polje kao relaciju prema lokaciji koja je sadržala oba objekta. Drugim rečima, ako imate potrebu da premestite dva polja da biste stvarno identifikovali slog, ali samo jedno od polja iz te tabele, koje je neophodno za izvršavanje pregleda tabele, potrebna Vam je nova tabela. Za baze podataka koje zadovoljavaju ovo pravilo se kaže da su u drugoj normalnoj formi.

U bazi Kadrovi, u tabelu Plate, radnik se ne unosi svojim imenom već identifikacijom preko njihovog ID broja (RadnikID). Na ovaj način se memorijski prostor za bazu uveliko smanjuje, održavanje baze se olakšava, a brzina operacija nad tabelama povećava.

PRIMER: Ako tabelu PLATE(RadnikID, Mesec, Sektor, Rukovodilac, LD) svedemo na PLATE(RadnikID, Mesec, SektorID, LD), a podatak o tome ko je Rukovodilac sektora definišemo u tabeli SEKTORI(SektorID, Sektor, Rukovodilac), tada će tabela PLATE biti u drugoj normalnoj formi.

Pravilo 3: Eliminisanje kolona koje ne zavise od primarnog ključa

Polje, koje nije u direktnoj vezi sa indeksnim ključem tabele, bi trebalo da bude preneto u drugu tabelu. U jednoj tabeli podataka bi trebalo da budu smešteni samo elementi podataka koji su u direktnoj relaciji sa primarnim ključem tabele. Ovo pravilo se najčešće odnosi na izvedene podatke ili druga izračunavanja.

Dakle, ako imate elemente podataka koji nisu u direktnoj relaciji sa primarnim ključem tabele, ove elemente bi trebalo da prebacite u njihove sopstvene tabele.

Drugim rečima, ako kolona tabele stvarno ne treba da bude u toj tabeli, ona je verovatno neophodna u nekoj drugoj tabeli. Za baze podataka koje slede ovo pravilo se kaže da su u trećoj normalnoj formi.

PRIMER: Ako tabelu PLATE(RadnikID, Mesec, Sektor, LD) svedemo na PLATE(RadnikID, Mesec, LD), a podatak o tome u kom je sektoru radnik unesemo u tabelu RADNICI imaćemo: RADNICI(RadnikID, Ime, Prezime, MB, Sektor)

Dok prva tri pravila normalizacije uključuju eliminaciju ponavljajućih, radudantnih ili relevantnih polja podataka, preostala dva pravila uključuju izolaciju višestrukih relacija u cilju poboljšanja opšteg integriteta podataka. Prva tri pravila su

obično sve što treba da ispoštujete da biste došli do dobro dizajnirane baze podataka, međutim, u slučaju kompleksnijih modela je neophodno slediti i sledeća pravila, a detaljno se obrađuju na višem kursu - *MS Access* (2).

Pravilo 4: Izolacija nezavisnih višestrukih relacija

Ne može postojati tabela koja sadrži dve ili više 1:n ili n:n relacija koje nisu u direktnoj vezi. Ovo pravilo se koristi za poboljšanje relacionog modela baze podataka kada on sadrži više od jedne 1:n relacije.

Drugim rečima, ukoliko je podatak važan, ali nije u direktnoj relaciji sa drugim elementima sloga, moraćete da prenesete ove podatke u novu tabelu. Za baze podataka koje slede ovo pravilo kaže se da su u četvrtoj normalnoj formi.

Pravilo 5: Izolacija zavisnih višestrukih relacija

Peto pravilo normalizacije podataka kaže, da je neophodno da izolujete nezavisne višestruke relacije u bazi podataka. Ovo pravilo se koristi za poboljšanje relacionog modela baze podataka kada on sadrži više od jedne n:n relacije.

Dakle, ako imate pravila koja zahtevaju višestruke reference prema istom polju ili setu polja, izolujte to polje u drugu tabelu i kreirajte jednu ili više tabela za povezivanje, koja će sadržati zahtevane reference (veze), a istovremeno zadržati integritet baze podataka.

Drugim rečima, ako nekoliko složenih relacija postoji u Vašoj bazi podataka, treba da izdvojite svaku od tih relacija u njihovu sopstvenu tabelu. Za baze podataka koje slede ovo pravilo kaže se da su u petoj normalnoj formi.

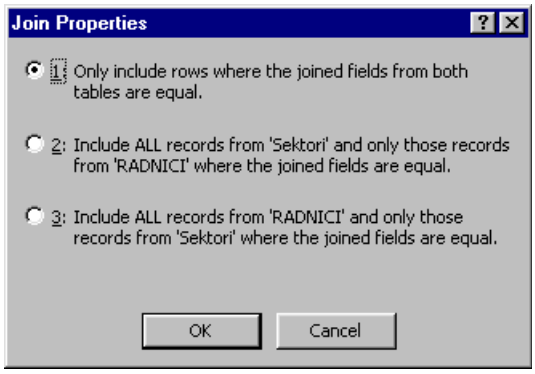
VEŽBA: Otvorite *MS Excel* fajl (roba.xls), pa prema podacima (kolonama u prvom radnom listu) kreirajte potrebne tabele u vašoj bazi (roba.mdb), s tim da baza bude normalizovana primenom pravila 1, 2 i 3.

7 RELACIONI MODEL

Izborom *Relationships* sa *Toolbar*-a *Database* otvara se prozor u kome se definiše relacioni model (veze polja između tabela). Iz dijaloga *Show Table* (koji se dobija sa *Toolbar*-a) biraju se tabele i upiti koji učestvuju u relacijama. Višestruko biranje objekata vrši se uz pomoć tastera *Shift*. Kada se tabele nađu na panelu, odgovarajuće relacije se kreiraju prenosom polja koje povezuju dva objekta od jednog do drugog. Spuštanjem na odgovarajuće polje u drugoj tabeli na panelu, pojavljuje se dijalog u kome se bira tip i osobine veze. *MS Access* će automatski prepoznati (sugerisati) vezu 1:N (*One-To-Many*) ili 1:1 (*One-To-One*). Ostaje još da se odredi da li treba uključiti referencijalni integritet i u okviru njega da li treba postaviti opcije za kaskadno brisanje i/ili kaskadno ažuriranje sadržaja tabela. Ostale tipovi veza mogu se definisati u *Join Type*.

Veza i relacija

VEZA je izraz za liniju koja spaja dva polja u dve tabele. Tabele se dakle vezuju preko polja podataka,a veza može biti i višestruka (preko više polja). Višestruke relacije mogu uticati na povećanje broja redudantnih (ponovljenih) podataka u bazi kao i na nemogućnost ostvarenja referencijalnog integriteta date relacije, pa ih treba izbegavati.



Ako veza između dve tabele nema oznaku tipa (1 i ∞) to znači da je veza tek inicirana.

RELACIJA predstavlja jednostruku ili višestruku vezu između dve tabele preko njihovih polja. Sve veze između dve tabele moraju da se uklapaju u jedan tip relacije (1:n, n:1, 1:1), što znači da se u jednoj relaciji ne mogu uključiti veze različitih tipova. Relacija se smatra definisanom (uspostavljenom) tek po uključenju referencijalnog integriteta (dijalog *Relationships*) i tipa relacije (dijalog *Join Properties*). *MS Access* prepoznaje tip relacije, pa se u ovaj dijalog ulazi da bi se tip relacije, eventualno, promenio (najčešće kod upita).

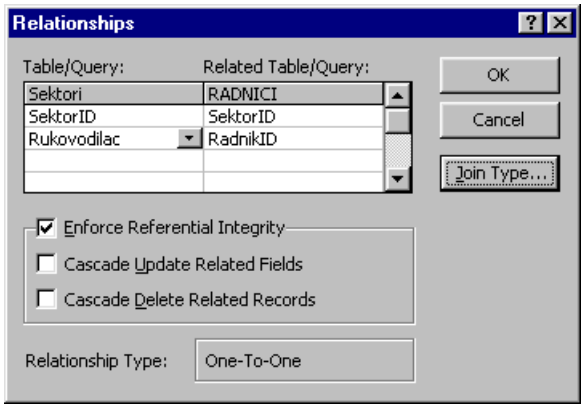
Relacije tipa 1:n i 1:1

Relacija tipa 1:n (ili n:1) je najčešći tip relacije, ali bi se slobodno moglo i reći da se dobar relacioni model zasniva samo na ovom tipu relacije između tabela.

Relacija tipa 1:1 je redi tip, jer daje mogućnost da se obe tabele (koje su u ovoj relaciji) smeste u jednu. U slučajevima kada u jednu tabelu treba smestiti veliki broj polja (atributi jednog entiteta) onda se, zbog preglednosti relacionog modela i njegovog lakšeg održavanja, polja "grupišu" u više tabela, a između definiše relacija 1:1. Kao primer za ovo se može uzeti neki kompleksan entitet kao što je avion, gde se on mora podeliti na grupe atributa (motor, putnički prostor, kabina posade, teretni prostor, ...), pa treba kreirati tabele prema ovim entitetima, a sve njih povezati po jednom polju (atributu) koji egzistira u svima (npr. AvionID).

Rekurzivna veza

Rekurzivnom vezama se nazivaju ciklične veze između dve ili više tabela. Ovo je slučaj kada bi se, praćenjem smera niza veza u smeru strelica (n:1), formirao kružni tok.



U slučaju dve tabele ovakva relacija se može inicirati, ali je uključivanje referencijalnog integriteta moguće samo u slučaju kada se ova veza zasniva na relaciji 1:1. Za primer se može uzeti relacija između tabela RADNICI(RadnikID, Prezime, Ime, SektorID) i SEKTORI(SektorID, Sektor, Rukovodilac), gde bi trebalo ove tabele vezati jednom vezom 1:n, a drugom n:1.

Drugi slučaj, kada rekurzivnu vezu treba ostvariti između više tabela (npr. tri tabele: RADNICI, SEKTORI, RUKOVODIOCI) je moguće ostvariti, ali se nameće problem održavanja referencijalnog integriteta u slučajevima kaskadnih popravaka i brisanja. Vrlo lako se može doći u situaciju da se brisanjem samo jednog sloga u jednoj tabeli izvrši ("nepotrebno") brisanje velikog broja slogova u druge dve tabele. Drugi problem ove veze je slučaj obavez-

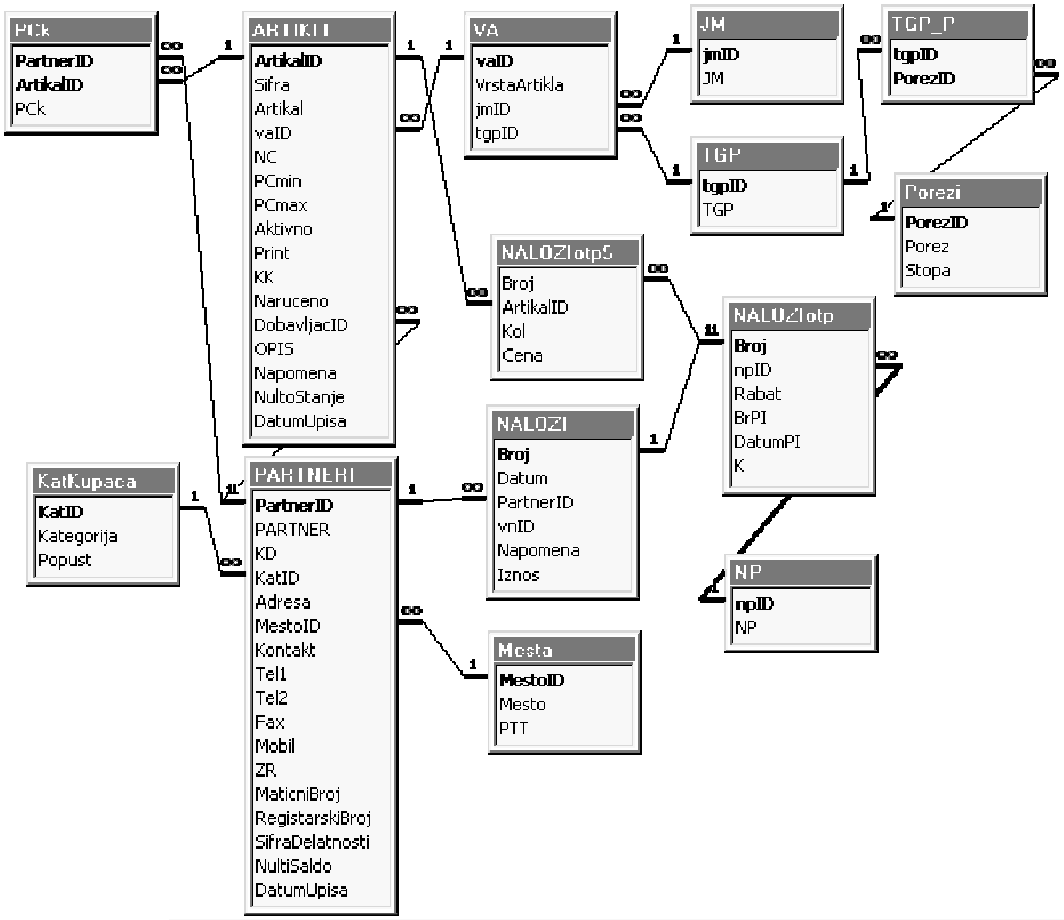
nog podatka koji se ne može upisati jer ne postoji u izvornoj tabeli, a koji na isti taj način ne može da se upiše zbog nedostatka izvornog podatka u trećoj tabeli.

Rekurzivna veza se može ostvariti, ali je treba izbegavati zbog nemogućnosti ostvarivanja ispravne relacije, odnosno njenog teškog održavanja.

NAPOMENA: Ako se ukaže potreba da se promeni primarni ključ u tabeli koja već učestvuje u relacijama, pre toga moraju se ukinuti sve njene veze sa ostalim tabelama. Brisanje tabela iz *Relationships* (panela sa relacijama) ne znači da su izbrisane i njene veze, već je ona samo sakrivena u cilju bolje preglednosti. Relacije ostaju čak i kada je tabela izbrisana sa panela, te ih treba brisati jednu po jednu, dok se veza briše – brisanjem vektora koji grafički povezuje tabele u *Relationships*.

Primer nešto složenijeg relacionog modela:

VEŽBA: Između tabela u bazi Kadrovi uvesti relacije kao na slici.



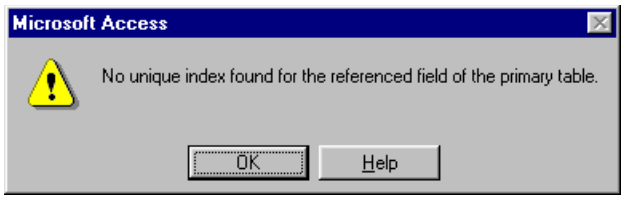
Problemi pri definisanju relacije

Relacija se ne može uključiti jer nije unapred zadovoljen referencijalni integritet, što se javlja u slučaju kada obrađuje model sa već postojećim podacima, pa referentno polje (destinacija ili izvor, a nalazi se na strani gde je i strelica) ne sadži podatak koji je upisan u zavisno (*Lookup*) polje (sa druge strane veze).

Referentno polje nije primarni ključ niti je indeksirano kao *No Duplicates*.

Referencijalni integritet

Referencijalni integritet (*Referential Integrity* - RI) se u određenoj relaciji može, ali i ne mora uključiti. Svaka relacija u kojoj RI nije uključen predstavlja potencijalnu opasnost za očuvanje ispravnosti rada baze. Ako u relaciji, RI nije uključen, tada se svi mogući slučajevi eventualnog ugrožavanja integriteta baze moraju programirati, što je daleko teže od uključivanja RI. Za sve greške i nastale probleme ugrožavanja RI, u 99% slučajeva, odgovornost je na administratoru baze podataka, dok se tek za 1% moge "okriviti": računar, *MS Access*, lokalna mreža, napajanje el. energijom i drugi faktori.



Problemi sa integritetom ažuriranja

Integritet kaskadnog ažuriranja treba uključiti u situaciji kada je ažuriranje izvornog podatka korisniku omogućeno. U bazi Kadrovi ova situacija se može nametnuti kada polje RadnikID u tabeli Radnici ne bi bilo tipa *Auto Number*. Vrednost u polju RadnikID je izvorni (referentni) podatak za mnoge tabele u bazi, pa ako bi se njegova vrednost menjala, trebalo bi, na isti način, ispraviti (*update*) i sva polja koja imaju tu vrednost. Dakle, ako bi se RadnikID u tabeli Radnici ispravio sa 3 na 333, to bi trebalo da se uradi i u tabelama Plate i P_R, a kako broj tabela može biti znatno veći, bolje je ovaj posao prepustiti kaskadnom ažuriranju *MS Access*-a. U slučaju većih zahvata na kaskadnom ažuriranju, prethodno treba kreirati rezervnu kopiju ili se pouzdati u neprekidno napajanje električnom energijom preko solidnog UPS-a.

Ako se kaskadno ili manuelno ažuriranje ne obavi u potpunosti i na kvalitetan način, doći će do situacije prividnog nepostojanja podataka. Pored toga što podataka nema tamo gde bi ih očekivali, oni se mogu pojaviti tamo gde ih ni slučajno ne bi očekivali, pa se ovo smatra najvećom mogućom greškom u održavanju baza podataka. Čak ni brisanje podataka nije tako opasno, jer se bar zna da ih nema, dok se u ovom slučaju javlja i problem njihovog nekontrolisanog "pojavljivanja" na pogrešnim mestima.

Problemi sa integritetom brisanja

Integritet kaskadnog brisanja se zahteva u slučaju da se izvorni podatak ne ispravlja već briše, jer u slučaju da se briše izvorni podataka, moraju se brisati i slogovi u drugim tabelama, čija polja se referenciraju na brisano polje (najčešće slog). Pretpostavimo da iz tabele Radnici obrišemo jednog radnika, tada bi se morali brisati slogovi tabela: Plate i P_R, u kojima se on pominje. Dakle, posle brisanja radnika sa podatkom RadnikID = 1, moraju se brisati i svi slogovi iz Plate i P_R u kojima je RadnikID = 1. U suprotnom bi došlo do situacije da polja i slogovi sa referencom ostanu bez izvora, a time i neupotrebljivi. Ovakva polja se mogu uočiti po tome što umesto očekivanog teksta sadrže samo broj (isprobati u bazi Kadrovi).

Rešenje normalizacijom

Primenom navedenih pravila normalizacije treba izbeći bilo kakav oblik redudanse (ponavljanja podataka na više mesta u bazi), ali i pravilno odabrati polja primarnog ključa. Savetuje se unos bar po pet slogova u svaku izvornu tabelu i bar 10-ak u tabelu sa referencom na izvornu, jer se ovim testiranjem praktično uočavaju načinjene greške u dizajnu tabela i relacionom modelu. Baze u kojima je primenjena normalizacija i u svim relacijama uključen referencijalni integritet, su znatno manje, a njihov rad brži i sigurniji. Ako se baza koristi u višekorisničkom okruženju, navedena pravila nisu dovoljna, pa je potrebno primeniti potpunu optimizaciju modela i radnog okruženja (server-računar, parametri lokalne mreže, Jet Engine, ODBC ili *SQL server*), što se obrađuje na višim kursevima.

VEŽBA: Na ovom terminu polaznici treba da izlože relacioni model baze podataka koju uzimaju kao svoj zadatak, a predavač treba da im sugeriše eventualne korekcije.