Windows baze podataka

Poput nekog velikog magneta, Windows efikasno "privlači" nove i nove aplikacije: najpre su mu se prilagodili grafički programi, onda tekst procesori a ovih dana Windows preotima i poslednji veliki bastion DOS-a, baze podataka. Rad u grafičkom okruženju oduvek je bio komforniji i prijatniji, a sada postaje i efikasniji...

Dejan Ristanović
Nenad Batoćanin

Korisnici "velikih" računara do skora su sa prilično prezirom gledali baze podataka na PC-ju: oduvek su bile šarenije, ali je opsežnija analiza otkrivala da je učinjen tek mali korak u razvoju ozbiljnog sistema zasnovanog na (toliko pominjanom) relacionom modelu. Time, naravno, nećemo da kažemo da su se PC-jeve baze podataka mogle koristiti samo za obradu kućnog telefonskog imenika ili inventara nekog magacina - moglo se isprogramirati sve što je potrebno, ali je programer bio suočen sa prilično opsežnim poslom, a korisnik koji zna o čemu se radi sa večitom neizvesnošću: ako je programer pogrešio u nekoj sitnici, podacima se crno piše!

Sigurni smo da su problem itekako osetili svi koji su koristili neku od starijih verzija *Clipper*-a za pisanje iole složenijeg programa. Nisu, možda, umeli da ga teorijski formulišu, ali im je bilo jasno kako se manifestuje: podaci se prostiru kroz nekoliko baza podataka, i dešava se da, zbog neke greške u programu, hardverskog ili softverskog problema (npr. nestanak struje) ili, u najgoroj varijanti, neke nepredviđene akcije korisnika, podaci u jednoj bazi ostanu neusklađeni sa podacima u drugoj. Obrišete, recimo, slog nekog kupca ali "zaboravite" da obrišete artikle koji su za njega rezervisani. Artikli se i dalje ne mogu koristiti, jer se, bar što se programa tiče, čuvaju za nekoga, a taj neko ne postoji! Često nam se dešavalo da dobra trećina *Clipper* koda koji napišemo ode na provere raznih neregularnih situacija - čim neki upis u bazu iz bilo kog razloga ne uspe, moraju se "odmotati" i prethodni uspešno obavljeni upisi kako bi podaci bili konzistentni. Najveći problem nastaje kada jedan upis ne uspe, pokušate da opozovete prethodno obavljene operacije a upisi i dalje ne uspevaju, obično zbog nekog hardverskog problema. Kada se problem otkloni, korisnik će morati da startuje (a vi ćete pre toga morati da napišete) program koji pregleda sve baze, pronalazi logičke nekonzistentnosti i onda ih ispravlja, automatski ili na osnovu uputstva operatera.

Zakon održanja baze

Mnogi *Clipper* programeri mislili su da tako naprosto mora da bude, ali će ih neki noviji paketi pokolebati u tom uverenju: već na prvi pogled videćete da *Microsoft Access* potpuno automatski održava konzistentnost podataka. I to je održava bez obzira na "trud" korisnika da tu konzistentnost poremeti: probali smo čak i da pritiskamo hardverski *reset* tokom (namerno izazvane) pauze koja prethodi upisu u neku od tabela, i sve je ostajalo u stanju u kome je bilo pre početka čitave operacije. Održanje konzistentnosti, doduše, zahteva neki minimum discipline od strane programera: treba dosledno koristiti mehanizam transakcija o kojima će biti reči kada budemo prikazivali *Access*. Ta sitna pažnja uložena u dizajn aplikacije višestruko će povećati njenu pouzdanost i učiniti čitav sistem daleko otpornijim na sve vrste problema.

Stručnim jezikom rečeno, nove baze podataka u značajnoj meri podržavaju relacioni model. To je formalni, matematički model baze podataka u kome se svi podaci vide samo kao skup tabela. Naziv "relacioni" potiče od toga što matematičke relacije mogu da se predstave tabelama, pa se, u stvari, u ovom modelu podaci vide kao skup relacija; može se reći da je relacioni model u suštini **način** na koji se vide podaci. Ovaj model obuhvata tri veoma važna elementa: podatke, pravila i operatore. Podaci čine isključivo skup relacija, odnosno tabela. Pravila obezbeđuju očuvanje integriteta podataka. Jedno pravilo kaže da se element relacije (tabele) mora razlikovati od ostalih - tabela mora imati tzv. **primarni ključ** na osnovu čije vrednosti se uvek jednoznačno pronalazi odgovarajući red (više o primarnom ključu docnije). Drugo pravilo govori o vezama između tabela. Neka se, na primer, za evidenciju poslovnih partnera u nekoj firmi uvedu tabele **Partneri** i **Gradovi**, pri čemu se za svakog partnera u tabeli **Partneri** navodi šifra odgovarajućeg grada. Ta šifra odgovara šifri grada u tabeli **Gradovi** i zove se *foreign key* (strani ključ). Pravilo integriteta nalaže da u tabeli partnera ne sme postojati šifra grada koji ne postoji u tabeli gradova. To znači da se ne sme dozvoliti brisanje grada ako ima podataka koji se "referišu" na njega.

Treći deo relacionog modela čini skup operatora koji služe za razne manipulacije sa podacima (tabelama). Tu su standardni skupovni operatori unija, presek, razlika, zatim specifične operacije kao što su projekcija i selekcija koje izdvajaju deo tabele i slično. Primenom ovih operatora na postojeće tabele dobijaju se nove.

Važnost relacionog modela najbolje ilustruje *C. J. Date* u poznatoj knjizi "*Database Systems*": relacioni model predstavlja mašinski jezik savremenih i budućih baza podataka! Relacione baze podataka su danas svakako preovlađujuće - uglavnom zbog jednostavne implementacije, velikih mogućnosti i jake formalne zasnovanosti. Povremeno se sreću sistemi zasnovani na drugim modelima, obično mrežnim i hijerarhijskim, a postoje i baze podataka sa mešovitim pristupom.

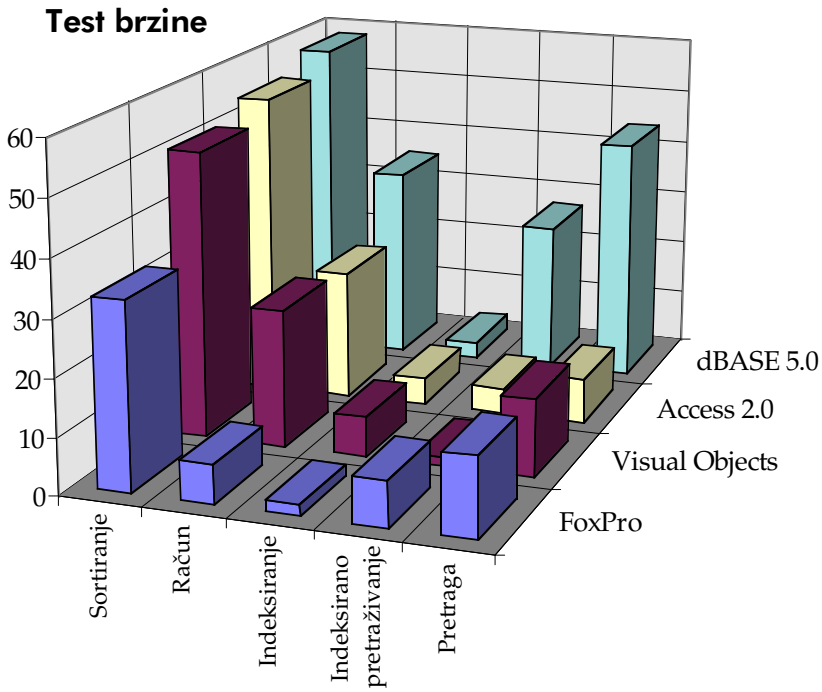
Kodov model

Treba reći da je gornji opis relacionog modela dat veoma grubo i u najkraćim crtama. Pitanje oko koga se veoma dugo lome koplja je "da li je neki sistem relacioni ili ne"? Postoje uslovi koje je formulisao *E. Codd* i koje treba da ispuni neka baza podataka da bi se nazvala "relacionom". Na žalost, ni jedna savremena baza podataka ne zadovoljava sva ta pravila. Ipak, na osnovu nekih pravila mogu se izdvojiti nekoliko grupa baza podataka koje "konkurišu" za naziv "relacioni sistem". Prvu grupu (koja zadovoljava najmanje pravila) čine tzv. **tabelarni** sistemi. U njima se svi podaci vide kao tabele, ali nema operatora koji operišu sa tim tabelama na nivou skupova. Drugu grupu čine tzv. **minimalni relacioni** sistemi koji imaju operatore selekcije, projekcije i spajanja tabela. U ovu grupu se mogu ubrojati i *xBase* proizvodi. Sledeću grupu čine baze koje podržavaju kompletniji skup operatora (*Oracle, Ingres*, DB2, ...). Zaključak ove priče je da *xBase* proizvodi u

suštini nisu pravi RDBMS sistemi, ali da po osnovnoj koncepciji spadaju u relacione baze podataka. Ono što im nedostaje je uglavnom veći skup operatora i nadgradnja nad osnovnim sistemom za upravljanje datotekama. Na primer, kod pravih relacionih sistema se pravila za očuvanje integriteta automatski "brinu" da *foreign key* u jednoj tabeli uvak ima ispravnu vrednost prilikom upisa, izmene ili brisanja podataka iz baze, dok se rad kod većine *xBase* proizvoda izvodi na mnogo nižem nivou: sve ove uslove mora da proverava program prilikom svake operacije.

Razlog za sve ove nedostatke je veoma jednostavan: do pre par godina PC računari nisu bili dovoljno moćni da bi se na njima vrtele baze podataka koje sve to omogućuju! Međutim, sada se situacija promenila: PC računari su po moći sasvim prestigli nekadašnje velike sisteme pa se neke tradicionalno "velike" baze podataka sada sele u "manja" okruženja (*Oracle*, na primer). Sa druge strane, *xBase* proizvodi dobijaju nove elemente koji im povećavaju moć i po karakteristikama ih sasvim približavaju "velikim" bazama podataka. Na primer, nove verzije *FoxPro*-a podržavaju SQL upite. Učvršćivanje pozicije *Windows*-a i masovna selidba aplikacija na novu platformu je ovom procesu dala nov kvalitet: prevazilaženje DOS barijera je doprinelo "unutrašnjem" kvalitetu baza podataka, a grafički korisnički interfejs je omogućio komforniji i atraktivniji spoljni izgled.

Što se popularnih proizvoda tiče, *Microsoft Access* je najpopularnija baza podataka koja se u velikoj meri prilagodila relacionom modelu. Na raspolaganju su mogućnosti očuvanja integriteta u okviru jedne tabele (*integrity enforcement*) - ako se uspostavi odnos između neka dva polja tabele, sam *Access* će brinuti o tome da taj odnos nikada ne bude narušen, bez obzira da li se podacima pristupa iz programa koji je posebno prilagođen toj bazi podataka, direktno iz *Access*-ovih menija, iz nekog posebnog *Windows* programa ili na bilo koji drugi način (naravno, ako vam ne padne da pamet da startujete *Norton Utilities* i pišete direktno po fajlu). *Borland Paradox* takođe uključuje sve bitne osobine ovoga tipa dok, recimo, *FoxPro* uopšte nema takvih mehanizama. Pojedini generatori aplikacija, kao što je *Clarion* ili *DataFlex*, nalaze se na polovini puta: ako kod generišete iz njih, integritet tabele se garantovano očuvava, ali ako samoj tabeli pristupite direktno ili iz nekog drugog programa, zaštite nema! Kod ostalih baza, integritet se očuvava isključivo kroz RANGE i VALID klauzule GET naredbi i njihove ekvivalente.



Kako smo testirali

Za detaljno testiranje odabrali smo četiri najveća i najznačajnija *Windows* RDBMS paketa - *Borland*-ov *dBASE for Windows 5.0*, *Visual Objects* firme *Computer Associates* (naslednik kod nas toliko popularnog *Clipper*-a), *Microsoft Access 2.0* i *FoxPro 2.6 for Windows* (nekada proizvod firme *Fox Software*, sada takođe *Microsoft*-ov paket). Za testiranje je korišćena mašina koju smatramo standardnom: 486/66 sa 256 K keša, 8 megabajta memorije i brzim diskom od 500 megabajta.

Benchmark testovi su, u odnosu na stara dobra DOS vremena, postali znatno ozbiljniji problem: *Windows* interno kešira rad, razlikuje se izvršavanje u pozadini i u "prednjem planu" i, uopšte, merenje je veoma osetljiva operacija u kojoj sitni propusti mogu proizvesti veoma neprijatne rezultate. Opredelili smo se da programe testiramo primenom posebno prilagođenog paketa domaće proizvodnje, koji se inače koristi u okviru *Unitest*-a za procenu performansi računarskih sistema i njihovih komponenti - rezultati su prikazani u pratećem dijagramu i sami po sebi su prilično ubedljivi; ovde ćemo samo opisati sam postupak testiranja i jedan ozbiljan problem na koji smo naišli.

Najpre se kreira baza od 10,000 slogova sa slučajnim sadržajem, a onda se mere parametri od značaja za rad sa podacima: pretraživanje (traži se pet razbacanih podataka koji postoje u bazi, a zatim jedan koji ne postoji; test se ponavlja 4 puta), računanje (sabiraju se vrednosti jednog numeričkog polja, kroz sve slogove baze, uz ponavljanje testa 12 puta), indeksiranje (formira se indeks čitave baze po jednom alfanumeričkom polju), indeksirano pretraživanje (traži se pet razbacanih podataka koji postoje u bazi, a zatim jedan koji ne postoji, i to 100 puta) i najzad sortiranje (po sadržaju alfanumeričkog polja, ponovljeno 4 puta). Vremena nismo uprosečili, tj. dajemo svako od njih posebno, pošto čitacima može biti od interesa da procenjuju one segmente rada koji su im u praksi najčešće potrebni.

Prilagođavanje istog programa za više različitih platformi uvek je složen posao, pre svega zbog toga što bi "mehaničko" prevođenje bilo nefer prema nekim modernijim programima. Uz razne probleme koje smo rešavali "u hodu", naišli smo na jedan koji se pokazao nerešivim: prilikom testa pretrage, u jednom od polja se javljao znak | koji, u poljima *Access* baze, označava sadržaj nekog polja tabele (npr. |[datum]|). Posle konsultovanja dokumentacije, *help*-a i raspitivanja po domaćim i stranim BBS-ovima, dolazimo do zaključka da se vertikalna crta sama po sebi nikako ne može tražiti, pa je baza podataka koju smo koristili za testiranje *Access*-a za bajt različita od baze na kojoj smo testirali ostale programe. To svakako nije uticalo na performanse, ali za čitaoce koji koriste YUSCII raspored ovaj momenat može da bude izuzetno neprijatan!

Šta izabrati?

Odluka o RDBMS sistemu koji ćete koristiti nije ni malo jednostavna. Najlakše bi bilo opredeljivati se na osnovu stvari koje se mogu egzaktno meriti, dakle performansi - teško da bi se tada neko mogao takmičiti sa paketom *FoxPro*. Za onoga ko se bavi razvojem RDBMS aplikacija druge stvari su, međutim, često važnije. Pre svega komfor pri radu i generisanje aplikacija koje će, zahvaljujući automatskoj proveri konzistencije, biti otpornije na sve vrste poremećaja. Od značaja su i veze sa drugim *Windows* programima, rad u mreži, a za ponekoga i klijent / server arhitektura odnosno ODBC kompatibilnost.

Pored svih ovih objektivnih kriterijuma, postoji i jako važno pitanje korišćenja postojećeg znanja i iskustva - za nekoga ko tek počinje, novi i moderni jezik je svakako nešto što se može preporučiti; za one koji za sobom imaju višegodišnje iskustvo u dBASE i *Clipper* programiranju, znatno je privlačnija mogućnost da nastave da koriste i unapređuju znanje koje su već stekli. Svakome je, naravno, jasno da se neke navike moraju menjati, ali je lakše kada to ide postepeno.

Naš izbor se, dakle, kreće u dva pravca. Oni koji žele da koriste modernu alatku koja u najvećoj meri prati relacioni model, obezbeđuje komforan razvoj aplikacija i podržana je od strane najveće svetske softverske kuće, mogu se slobodno opredeliti za *Microsoft Access* 2.0. Za one koji su bliži "klasičnom" programerskom razmišljanju i koji bi postepeno ulazili u svet objekata koristeći stečena znanja i (što je posebno važno) portirajući postojeći softver ne novu platformu, dobar izbor bi mogao da bude paket *Visual Objects* firme *Computer Associates*. *dBASE for Windows* je zanimljiv pre svega za one koji samostalno razvijaju baze podataka koje će docnije koristiti, ali nepostojanje *Developers Pack*-a smanjuje interes programera koji bi se bavili razvojem softvera za tržište. *FoxPro* za *Windows* bi, da je nastavila da ga razvija nezavisna firma, mogao biti veoma opasan takmac paketu *Visual Objects* (a pogotovu dBASE-u za *Windows*) ali je, u rukama *Microsoft*-a, budućnost ovoga paketa pod velikim znakom pitanja. *PowerBuilder, Clarion, DataEase, DataFlex, R:BASE* i neki drugi programi koje kod nas retko ko koristi svakako su vredni daljeg istraživanja.

Access je pri vrhu i po pitanju očuvanja konzistentnosti podataka u raznim tabelama (*referential integrity*): operacija koja bi štetila sinhronizaciji sadržaja nekoga polja sa sadržajem polja u drugim tabelama jednostavno biva odbijena. U literaturi ćete ovu koncepciju često sresti pod imenom *reject operations*: sama baza podataka **odbija** izvršavanje nekih komandi! Drugi autori koriste termin *restrict operations*, u smislu da su neke operacije zabranjene. Osim *Access*-a, ove mehanizme podržavaju i *Borland Paradox for Windows* (ali **ne** i *Paradox for DOS*!) i R:BASE. Druge baze podataka obezbeđuju neke od ovih funkcija, npr. kaskadno brisanje slogova, ali je u većini slučajeva programeru ostavljeno da brine o tim "sitnicama".

Ostaje još da detaljnije opišemo već pomenuti princip primarnog ključa koji dBASE, sticajem istorijskih okolnosti, nikada nije podržavao. Šteta: primarni ključ je izvanredan način da baza podataka ne bude zavisna od fizičkog redosleda kojim su podaci upisani na disku. U dBASE-olikim jezicima se, kao što znamo, operiše pojmom "redni broj sloga": do željenog podatka u bazi se dolazi komandom GOTO, uz navođenje njegovog apsolutnog broja. Jednostavno za realizaciju, ali uz razne nedostatke: ne može se, recimo, ubaciti slog "u sredinu" baze bez njenog kompletnog prepakivanja. Brisanje podataka takođe mora da se proprati prepakivanjem, ako ne trenutnim a ono odloženim (komanda PACK). Po sortiranju će redni brojevi biti potpuno izmešani, što znači da veze dve baze na osnovu rednog broja sloga jednostavno ne dolaze u obzir. Što je najneprijatnije, naredba GOTO uopšte ne zavisi od eventualnog aktivnog indeksa: GOTO 1000 će aktivirati hiljaditi fizički slog baze, bez obzira što bi, po aktivnom redosledu koji indeks diktira, taj slog možda bio prvi ili poslednji. Naredba GOTO baš nema sreće: u *Clipper*-u je, doduše, oslobođena značenja koje ju je učinilo odgovornom za "špageti" programiranje, ali joj je ostala možda i štetnija uloga logički nekontrolisanog kretanja po bazi podataka!

Od ključa do indeksa

Primarni ključ može da se zamisli kao poseban indeks koji se implicitno formira: jedno polje baze se proglašava za ključno i na osnovu njega se uvek (i trenutno) pronalazi željeni slog. Podrazumeva se da ne smeju postojati dva polja koja imaju isti primarni ključ, ali što se realizacije tiče postoje dve varijante: većina RDBMS programa automatski proverava jedinstvenost ključa i odbija upis sloga sa dupliranim primarnim ključem. Takvi su, recimo, *Access, Paradox* i R:BASE. Štaviše, *Access* i R:BASE omogućavaju automatsko generisanje primarnog ključa kroz autoinkrementna polja: sama baza će za svaki novi slog formirati jedinstveni ključ. Neki drugi programi, na primer *DataEase*, zahtevaju od programera da vodi računa o konzistenciji primarnog ključa, pružajući mu neke opcije koje olakšavaju te provere. *FoxPro* i dalje ne podržava ovaj koncept.

Na primarni ključ logično se nadovezuje pojam indeksa: za brzo pronalaženje željenih podataka treba dobro organizovati "mrežu" polja koja se indeksiraju. Svi RDMBS programi sa kojima smo radili indekse podržavaju na relativno sličan način - formira se poseban fajl sa ključnim vrednostima nekog polja (ili izraza zasnovanog na vrednostima nekoliko polja) i fizičkim pozicijama odgovarajućih slogova a onda se, primenom posebnih naredbi, pokazivač pozicionira na slog sa zadatom vrednošću, ili se nekom formom SKIP komande pomera po bazi prema logičkom redosledu zasnovanom na indeksu. Razlike se ispoljavaju pre svega u stepenu automatizacije čitavog mehanizma: u dBASE-u i "srodnim" dijalektima programer raspolaže posebnim komandama koje operišu sa indeksima, i indeksi su stalno u centru njegove pažnje. U *Access*-u se indeksi, gledano spolja, jedva i vide: setite ih se samo kada kreirate bazu podataka, a ubuduće ih potpuno samostalno održava sam RDMBS, aktivirajući po svakom korisnikovom upitu onaj indeks koji će ga na najbrži način zadovoljiti. Koliko god ovaj mehanizam izgledao savršeniji, u praksi pokazuje i određene mane, pre svega zato što korisnik nije svesan koji će se upit razrešiti preko indeksa a koji će zahtevati pretraživanje čitave baze. To se pretraživanje, naravno, ponekad ne može izbeći, ali je većina takvih upita, uz malo više razmišljanja, mogla da se svede na daleko bržu pretragu po indeksu. Na neki duži rok gledano, naravno da automatski rad predstavlja varijantu koja dobija: korisnik zna koji ga podatak ili izveštaj interesuje, i savršeno ga se ne tiče šta se "unutra" dešava da bi se taj podatak dobio! Međutim, mnogi korisnik bi, u sadašnjoj fazi razvoja RDBMS programa, ipak bio voljan da malo razmisli o indeksima i sličnim tehnikalijama, i da na osnovu toga dobije podatak odmah, a ne posle pet minuta čekanja.

U konkretnim implementacijama rada sa indeksima treba proučiti još dva važna pitanja. Prvo se odnosi na smeštanje indeksa na disk: stari dBASE III je promovisao

(naoko logičnu) konvenciju da se svaki indeks upisuje u poseban fajl. Kako su diskovi rasli a baze podataka postajale složenije, broj fajlova se povećavao što je otežavalo i usporavalo rad, pa je u *FoxPro*-u uveden novi "zajednički" indeks koji obuhvata sve indekse pridružene nekoj bazi podataka. *Access* je otišao još korak dalje pa je sve objedinjeno u jedan fajl: podaci, maske, programi, indeksi... Iako ovaj pristup nije lišen privlačnosti, pre svega u smislu lakšeg manipulisanja čitavim projektom i efikasnijeg prenošenja na drugi računar, nismo sigurni da on predstavlja korak na dobru stranu: uvek se mora raditi pun *backup* podataka koji nisu menjani, otežan je (ili barem usporen) pregled programa u posebnim editorima i njihovo štampanje na način koji programeru najviše odgovara, sve je "zatvoreno" u integrisano okruženje, a postavlja se i pitanje kojim softverskim alatima "opraviti" taj mega-fajl, ako bude fizički oštećen. Bilo kako bilo, trend je očit: sve što čini jednu bazu podataka polako se ujedinjava u što manji broj datoteka!

Drugo važno pitanje odnosi se na YU slova: nikako da prevaziđemo vreme *patch*-ovanja raznih LIB-ova kako bi se reči sortirale po redosledu koji nam je potreban! *Windows*, teorijski posmatrano, može da se prilagodi našem jeziku: postoji verzija za Istočnu Evropu koja nam je u startu prilagođena, a i američka verzija se relativno lako "pripitomi" primenom "Srpskog jezičkog drajvera" (SLD) koji je napisao Obrad Bijelić. Na žalost, višejezičke mogućnosti *Windows*-a su od pomoći samo ako ih autori programa podržavaju, odnosno ako porede stringove pozivom funkcija koje su namenjene tom poređenju. Pokazuje se da ni sam *Microsoft* nije sledio sopstvenu konvenciju, tako da je SLD od slabe pomoći kada se radi sa *Access*-om; možete onda misliti tako stoje stvari sa drugim proizvođačima. Ukratko, ispravno domaće sortiranje i dalje se svodi na klasično rešenje, uz primenu neke od formi međukoda.

Priča o indeksima nije samo priča o klasičnim rešenjima: pre godinu ili dve se pojavila jedna blistava novost pod pomalo neobičnim imenom: *Rushmore* optimizacija. Doneo ju je, i za sada je jedini u punoj meri primenjuje, *FoxPro*: *Microsoft* je, možda baš zbog ove optimizacije, kupio firmu *Fox Software* i deo ove tehnologije već ugrađuje u *Access 2.0*, sa tendencijom da je podrži u celini. Ostali proizvođači kao da još razmišlaju o nečem sličnom.

Moramo da priznamo da smo, pri prvom kontaktu sa *Rashmore* tehnologijom koja je primenjena u *FoxPro 2.0*, čitavu stvar shvatili pre kao reklamni trik nego kao suštinsku novost: izgledalo je da se slične stvari mogu postići i na druge načine. Pokazalo se, međutim, da *Rushmore* predstavlja po svemu novu ideju, bar u svetu baza podataka za PC računare: umesto da se pri pretraživanju koristi jedan određeni indeks, istovremeno se konsultuje njih više! Zapravo, upit korisnika se automatski raščlani tako da se svaka od komponenti pretražuje po odgovarajućem indeksu, ako taj indeks postoji. Naravno, delovi upita koji se ne oslanjaju ni na jedan indeks se i dalje moraju rešavati sekvencijalnom pretragom, ali se na osnovu "indeksiranog dela" ta pretraga svodi na ispitivanje minimalnog broja slogova. Neoprezno zadat upit i dalje će zahtevati pregled čitave baze, što znači da je za efikasnu primenu *Rushmore* tehnologije i dalje potrebna određena saradnja korisnika, ali se u zbiru posao obavlja primetno brže nego u drugim dijalektima.

Upiti, koje smo upravo pomenuli, predstavljaju jednu od oblasti u kojima su RDBMS programi u poslednje vreme bitno uznapredovali. Pre samo par godina dBASE IV uopšte nije podržavao nikakvu pažnje vrednu formu SQL-a (*Select Query Language*, programski jezik za postavljanje upita). Danas dBASE V, *Access* i praktično svi ostali jezici uključuju veoma razvijen, premda međusobno slabo kompatibilan, SQL. Dobar SQL je ujedno i ulaznica u svet klijent - server aplikacija: *Paradox* i *Access* su tu otišli najdalje, pošto prvi uključuje *Borland SQL Link* drajver koji obezbeđuje vezu sa *Borland, Microsoft, Sybase* i *Oracle* bazama, dok drugi sarađuje sa svim ODBC (*Open Database Connectivity*) kompatibilnim bazama. *FoxPro* za DOS i *Windows* u osnovnoj verziji ne podržavaju ovakav tip "veze sa spoljnim svetom", ali se dobijaju biblioteke čijim se uključivanjem u kod ostvaruje ODBC kompatibilnost.

Savršeno okruženje

Iako smo ovaj uporedni prikaz počeli poređenjem koncepcije i tehničkih svojstava raznih RDBMS programa, pitanje korisničkog interfejsa je ono što privlači pažnju većine početnika u ovome svetu. Možda nije toliko bitno (mada nije ni nevažno) kako program izgleda, ali je od ključnog značaja koliko ćete vremena morati da uložite u njegovo proučavanje i od kolike će vam pomoći pri rešavanju konkretnih problema biti ugrađena automatika.

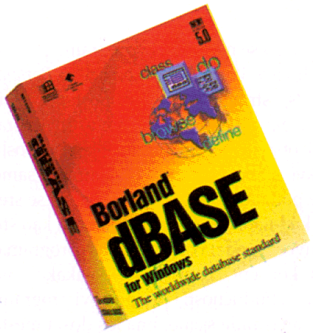
Autori *Windows* baza podataka mnogo rade na ovim pitanjima, a ima su utisak da su najdalje otišli *Borland* i *Microsoft*. Obe velike firme kreću se sličnim putevima: ideja je da se napravi niz programa (*Microsoft* ih zove *Wizards*, "čarobnjaci", a *Borland Experts*, "stručnjaci") koji će postaviti korisniku niz pitanja i, na osnovu odgovora, sami obaviti željeni posao. Najviše posla ovakve alatke obave pri dizajniranju same baze podataka, maski i izveštaja, ali se sreću i kod naoko trivijalnih operacija kao što je "uvoz" podataka iz drugih programa. Iako koncept "čarobnjaka" svakako ima blistavu budućnosti i premda ovi programi već danas rade sjajno, ima još dosta mesta za njihova poboljšanja, proširenja i, uopšte, prilagođenja ukusima i potrebama ne samo početnika nego i iskusnijih korisnika RDBMS programa.

Što se grafičkog izgleda tiče, sam *Windows* diktira standarde kojih se većina softverskih firmi drži, pa svi programi koje prikazujemo izgledaju relativno slično. Iz verzije u verziju jača tendencija koja je prisutna još od dBASE-a III a koja je "osnažena" u *FoxPro*-u: integrisano okruženje u kome će se raditi čitav posao, počev od dizajniranja same baze, maski i drugih resursa, preko unosa podataka i generisanja izveštaja, pa do pisanja, prevođenja, testiranja i prilagođavanja programa i makroa. Razlike se vide tek kada se zaviri "ispod haube": *Access* je promovisao sasvim novi, dosledno objektno orijentisani, jezik za razvoj *data base* aplikacija, dok *Borland* i *Computer Associates* i dalje osavremenjavaju jezike koji vode poreklo od dBASE-a, naravno uz brojne objektne ekstenzije i prilagođenja grafičkom okruženju.

Borland dBASE for Windows

Nema korisnika koji radi sa bazama podataka a da nije čuo (a i "opipao") čuveni dBASE III+ - godiname je to bila osnovna referenca! Često je umesto njega korišćen *Clipper* ili *FoxPro*, ali sve se uvek "merilo" prema dBASE-u. Onda je izašao dBASE IV koji očito nije bio korak u dobrom pravcu: tavorio je i tavorio na tržištu, ali nikada nije postigao naročitu popularnost (o popularnosti na našim meridijanima ne vredi ni spominjati - dBASE IV praktično niko nije ni koristio)! *Ashton Tate* nije mogao dugo da izlazi na kraj sa takvim poslovnim "uspesima" pa je firma prešla u ruke daleko većeg *Borland International*-a.

Prvi *Borland*-ov korak na oživljavanju dBASE standarda je verzija 5.0, za DOS i za *Windows*; ovde nas prevashodno zanima ova druga. Dizajniranje nove verzije očito je započeto od samih "temelja", tj. od takozvane *Borland Database Engine* biblioteke koja pre svega operiše sa podacima, ali i obezbeđuje vezu sa svim ODBC kompati-



bilnim okruženjima (*Oracle 6, Paradox, SQL server...*). Na tu "mašinu" nakalemljen je programski jezik koji je po sintaksi sličan dobro poznatom dBASE-u, ali je dopunjen brojnim objektnim ekstenzijama. Objekti se kreiraju komandama NEW (na "C++ način") ili DEFINE ("dBASE način"), a metodi preko takozvanog funkcionalnog tipa koji zapravo predstavlja pointer na funkciju - dodeljivanjima tipa *objekat.metod=funkcija* objektima se mogu pridružiti metode, što je ključ za (delimično) učaurivanje. Treba, ipak, primetiti da dBASE-ovi objekti, premda opšte prisutni, nisu tako razvijeni kao, recimo, *Clipper*-ovi: kontrola grešaka svodi se na ON ERROR komandu, a poređenje fleksibilnosti *Clipper*-ovog BROWSE objekta sa dBASE-ovim EDIT i BROWSE naprosto nema smisla, naravno u korist *Clipper*-a. Zapravo, ima se utisak da je *Borland* ove osetljive segmente pre zasnovao na procedurama nego na objektima.

Mnogo veći problem je u tome što dBASE i dalje ne podržava deklarisanje tipova, što u rad sa objektima uvodi neprihvatljiv element proizvoljnosti i povećava rizik od grešaka koje se vrlo teško lociraju. Objekti su, osim toga, uglavnom usmereni prema kreiranju grafičkog interfejsa - u radu sa bazom podataka ne nalaze svoje mesto, mada korisnik, naravno, može da kreira sopstvene objekte koji bi se bavili ovim važnim segmentom.

Važna novost dBASE 5.0 programskog jezika su kodni blokovi, koji su pre par godina obogatili i *Clipper* okruženje. Postoji, ipak, i jedna važna prednost na strani dBASE-a: dok u *Clipper*-u kodni blok u osnovi predstavlja izraz koji se izračunava, *dBASE for Windows* poznaje kodne blokove u koje su upisani čitavi programi: nije posebno bitno da rezultat toga bude broj, a u kodni blok ulaze i potpune komande tipa USE, REPLACE i slične. Mogući kodni blok bi, dakle, glasio: {|n, a| ; for i=1 to n; ? a; next i}. Pomenimo i impresivan spisak ugrađenih funkcija i procedura: gotovo da je dostignut u ovom domenu legendarni *FoxPro*, koji nudi po funkciju za svaku zamislivu radnju.

Iako zasnovan na relacionom modelu, dBASE 5.0 nije naročito napredovao prema zadovoljenju kriterijuma opisanim u uvodnom tekstu. Uvedene su, doduše, neke naredbe koje štite integritet podataka u raznim bazama (SET RELATION CONSTRAIN i INTEGRITY), ali sve to deluje prilično amaterski u odnosu na mogućnosti *Microsoft Access*-a.

Indeksi su potpuno reorganizovani, možda po ugledu na *FoxPro*: u prethodnim verzijama indeks po svakom od polja upisivan je u poseban fajl sa ekstenzijom NDX, a sada su svi indeksi u zajedničkom fajlu tipa MDX. Ovi indeksi se ne moraju posebno aktivirati - izborom tabele automatski se aktiviraju i indeksi koji joj pripadaju. Stari format indeksa podržan je zbog kompatibilnosti, ali se ne preporučuje njegova primena. Efikasnost rada sa indeksima je osetno unapređena u odnosu na dBASE III+, ali i dalje nije ravna onome što *Fox* pruža.

Neke od starih boljki dBASE-a pri radu sa indeksima su, na žalost, i dalje prisutne: bilo kakav REPLACE polja u glavnom indeksu izaziva neurotično ponašanje, a često i totalni krah programa. Osim toga, iako indeksi mogu da se formiraju po vrednosti nekog izraza, sortiranje je i dalje ograničeno na sadržaj polja što znači da će se indeksi, kao i do sada, koristiti i za ono što im nije primarna namena.

dBASE-ov korisnički interfejs deluje izuzetno profesionalno, a opet toliko jednostavno za upotrebu da će se dobar deo početnika snaći i bez čitanja dokumentacije ili potrage po *help*-u. Važne novosti na tom planu su *expression builder*, u kome se pomoću miša zadaju čitavi izrazi (mi preferiramo da ih otkucamo, ali....) i *SafetyNet* koji u startu onemogućava greške u izrazima. Naročito je impresivan *form designer*, podsistem u kome se crtaju maske za unos podataka - uz sve lepote, možete da napravite "podformular" (*child form*), recimo u obliku *browse* prozora na drugu bazu u okviru sloga u prvoj. *Form designer* nije samo generator korisničkog interfejsa - on je i generator programa! Sve ono što rasporedite po ekranu pretvara se u program u dBASE jeziku koji se automatski upisuje u datoteku sa ekstenzijom WFM. Nju možete koristiti na dva načina: da bolje upoznate objektno orijentisani programski jezik ugrađen u *dBASE for Windows*, ili da kreirate kostur programa a onda sami dodajete izvršne segmente. Ovaj drugi metod je zapravo najefikasniji način da dođete do aplikacije - oslonite se na *form designer*, ne trudeći se da razumete kako rade i šta znače svi oni *click event*-ovi koje je ugradio u program, a onda na sasvim klasičnom jeziku dodajte na potrebna mesta segmente koda koji će, na osnovu zadatih ulaza, pregledati bazu podataka, ažurirati polja i generisati izveštaje. Ovom koncepcijom *Borland* se priključuje trendu "vizuelnih jezika" kod kojih se mišem programira bar onoliko koliko i tastaturom.

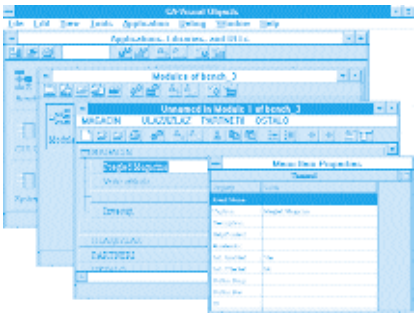
Finalna dorada programa je, na žalost, ostavljena za neko drugo vreme ili za neku drugu firmu - ni pola godine po izlasku na tržište, *Borland* nije dovršio razvoj *Distribution Kit*-a koji bi omogućio da se čitava aplikacija prevede i rasporedi na instalacione diskete koje biste kasnije mogli da prodajete. To praktično znači da svaki korisnik koji želi da radi sa dBASE 5 programima mora nabaviti kompletan dBASE 5 paket - veoma profitabilno za *Borland* ukoliko se mnogo njih odluči na kupovinu, ali se bojimo da je konkurencija na tržištu takva da će oni koji rade razvoj naprosto "otići u drugu firmu". dBASE 5.0 tako ostaje sjajna DBMS alatka, posebno pogodna za one koji dobro poznaju xBASE jezike i nemaju preteranu želju da uče nešto novo, ali će oni koji rade na razvoju dBASE aplikacija za tržište verovatno biti veoma retki!

Computer Associates Visual Objects

PC Preporuka

Firma *Computer Associates* ima iza sebe bogatu tradiciju u izradi baza podataka. *Visual Objects* je plod ambicioznog projekta i kupovine nekoliko softverskih firmi. Presudni uticaj na izgled novog proizvoda dao je kod nas jako popularni *Nantucket Clipper*; u stvari, pri projektovanju *Visual Objects*-a ključni zadatak je bio očuvanje kompatibilnosti sa *Clipper*-om 5.2, što je u velikoj meri i ostvareno.

Visual Objects ovaj prelaz omogućava u više nivoa. Najniži je terminal emulacija: *Clipper* programi, uz zaista minorne intervencije, mogu da se prebace direktno u *Visual Objects*. Glavni problem te transformacije je korisnički interfejs, a upravo o njemu se "brine" terminal emulacija: kada se startuje program koji radi u "DOS" modu, otvara se tzv. *terminal window* koji emulira DOS ekran: sve standardne *Clipper* naredbe rade potpuno isto kao i u DOS-u, uz zgodan dodatak automatske kontrole miša. Naravno, ovakve aplikacije su vizuelno prilično "jadne" u okruženju kao što je *Windows*, pa je ovo predviđeno samo kao prelazni korak ka pravom *Windows* programu. Sledeći korak je postepeno obogaćivanje programa *Windows* elementima. Tako postoji funkcija *CRTAddMenu* koja služi za re-



alizaciju *Windows* menija, *CRTAddButton* za "tastere" i slično; ovako dobijeni programi imaju korisnički interfejs koji samo "liči" na *Windows*. Sledeća faza bi bila transformacija u pravi *Windows* SDI (*Single Document Interface*) ili MDI (*Multiple Document Interface*) korisnički interfejs. *Visual Objects* raspolaže čitavim arsenalom alata i gotovih klasa koje pomažu pri kreiranju složenih elemenata *Windows* interfejsa. Na žalost, za ovaj korak je već potrebna korenita izmena u strukturi postojećih programa, zbog potpuno različite filozofije *Windows* i DOS platforme. Veoma je ohrabrujuća vest iz firme CA da se u bliskoj budućnosti planiraju verzije *Visual Objects* koje bi radile na drugim grafičkim operativnim sistemima kao što su *OS/2, Windows NT, Motif*...

Jedan od glavnih kvaliteta *Visual Objects*-a je nasleđen iz *Clipper*-a: programski jezik je izuzetno razvijen, dodato je mnogo novih funkcija i naredbi, a glavna novost je puna objektna implementacija. Sada korisnik ima mogućnost kreiranja sopstvenih klasa, što nije bilo moguće u *Clipper*-u. *Visual Objects* ima u sebi dosta ugrađenih klasa koje predstavljaju vrh hijerahije klasa za rad sa bazom podataka, korisnički interfejs i slično. Programski jezik je doživeo brojna i značajna poboljšanja. Na primer, sada su osim standardnih polimorfnih promenljivih koje nemaju fiksiran tip podržane i strogo tipizirane promenljive čiji je tip određen za vreme prevođenja. Time se izbegavaju greške koje nastaju zbog osobine polimorfnih promenljivih da prihvataju razne tipove podataka.

Za razliku od sličnih konkurentskih proizvoda, *Visual Objects* je **pravi** prevodilac: program se prevodi u mašinski kod koji se izvršava gotovo jednako brzo kao i neki program pisan u C-u, što naročito dobro dođe kod analize stringova, operacija sa nizovima i tome slično. U prevodilac je ugrađen i optimizator koda koji ima nekoliko nivoa optimizacije. Zadržana je i mogućnost generisanja finalnog EXE fajla koji će funkcionisati potpuno nezavisno od osnovnog *Visual Objects* okruženja i raznih *run time* modula. Naravno, distribucija ovako generisanih programa kao i u slučaju *Clipper*-a ne podleže nikakvim ograničenjima - kupite *Visual Objects* i možete da pišete i prodajete programe!

Pisanje programa u *Visual Objects* je veoma lepo rešeno. Mogu se pisati klasični programi, biblioteke ili DLL biblioteke - u terminologiji *Visual Objects*-a sve ovo se naziva "aplikacija". Svaka aplikacija se sastoji od elemenata - entiteta koji mogu biti procedure, funkcije, definicije, ali i meni, maska, itd. Ukratko, entiteti su svi elementi koji čine aplikaciju - pisanje programa se sastoji od dodavanja novih entiteta. Radi lakšeg snalaženja, svi entiteti se dele na module, slično deljenju velikog projekta na više datoteka sa izvornim kodom. Međutim, *Visual Objects* ne poznaje pojam "datoteka sa izvornim kodom": svi entiteti se čuvaju u posebnom sistemskom rečniku pod potpuno automatskom kontrolom *Visual Objects*-a: programer o tome uopšte ne vodi računa. Na primer, kada se dodaje nova procedura, dovoljno je da se u editoru napiše njen kod: *Visual Objects* je automatski snima i sve operacije sa datotekama su potpuno transparentne za korisnika. Da stvar bude još lepša, ne postoji razlika u tretmanu entiteta (na primer, menija i procedure): kada se definiše meni (možda u istom modulu kao i procedura ili izveštaj), sistem ga automatski snima i tu prestaje briga korisnika.

Uopšte, svi entiteti se kreiraju specijalnim alatkama: izvorni kod se kreira editorom, meniji se kreiraju editorom menija, prozori editorom prozora i slično. Kreirani entitet se upisuje u rečnik, i ni po čemu se ne razlikuje od ostalih entiteta. Kasnije izmene nekog entiteta automatski startuju odgovarajuću alatku: ako menjate neku proceduru, startovaće se "običan" editor, ali ako zatražite izmenu menija, startovaće se editor menija. Ovakvo "objektno" tretiranje svih entiteta veoma pojednostavljuje organizaciju projekta i značajno olakšava rad. Same alatke su urađene u skladu sa zahtevima *Windows* okruženja: rad je intuitivan i relativno jednostavan. Za program se koristi *source code editor*, za korisnički interfejs *window* i *menu* editori, *report* editor generiše izveštaje, *icon* editor ikone, a postoje i specijalni editori za definisanje baze podataka.

Prednost ovakve organizacije se vidi i prilikom prevođenja aplikacije. Naime, integrisana *Visual Objects* okolina prilikom svake izmene prevodi samo one entitete koji su menjani. To znači, da ako je menjana samo jedna kratka procedura iz nekog modula, neće biti prevođen kod čitavog modula (što je uobičajeno kod DOS aplikacija), već samo ta procedura. Ovo drastično ubrzava *edit-compile* ciklus, a samim tim skraćuje vreme razvoja.

Visual Objects može pristupati bazi podataka na dva načina: prvi je standardni način preko RDD drajvera nasleđen iz *Clipper*-a, a drugi način je korišćenjem standardnog ODBC protokola za pristup drugim bazama podataka. Korišćenjem ovog protokola (i uz odgovarajući drajver) može se pristupati svim SQL (i ne-SQL) bazama podataka koje podržavaju ODBC protokol.

Od RDD drajvera podržani su standardni DBFNTX, DBFCDX i DBFMDX. CDX drajver je kombinacija poznatih *Comix RDD*-a i *FlexFile* podrške za memo polja. *Comix* drajver omogućuje automatsku optimizaciju operacija tipa SET FILTER korišćenjem otvorenih indeksa. Ovim se operacije SET FILTER, LOCATE i sl. mogu ubrzati za red veličine. Sastavni deo DBFCDX drajvera je i podrška za BLOB polja; dudeći po iskustvima iz *Clipper*-a, ova kombinacija će se veoma često koristiti!

Bazi podataka se može pristupati na tradicionalni način kao i u svim *xBase* proizvodima: naredbe USE, REPLACE, INDEX, SEEK imaju potpuno isto dejstvo. Postoji i alternativni (preporučljiviji) način koji koristi tzv. *data server* objekte - radi se o posebnim objektima preko kojih se pristupa podacima u određenoj tabeli. Podacima se pristupa preko *instance* promenljivih, pa je sama fizička struktura podataka potpuno nebitna za rad: iza *data server* objekta može se kriti DBF datoteka, SQL tabela ili nešto treće, za program je nebitno. *Data server* rešava i problem virtuelnih polja, otvaranja pridruženih indeksa i slično. Na žalost, *Visual Objects* ne poseduje nikakav transakcioni mehanizam, pa je obrada transakcija i dalje isključiv zadatak programera.

Osnova poslovne politike koju je CA primenjivao na *Clipper* ostaje na snazi: veliki (možda i preveliki) prostor je ostavljen za nezavisne proizvođače raznih dodatnih proizvoda; bogatstvo ovakvih proizvoda je činilo veliki deo snage *Clipper*-a. Bez obzira što je *Visual Objects* relativno nov proizvod, za njega se već pojavilo puno dodataka, počev od unapređenja rada sa bazom podataka do dodataka za stvaranje multimedijalnih aplikacija. U spisku dodatnih proizvoda se prepoznaju poznata imena: *SIx* drajver, *CLGraph, CLTools, Flipper*...

Prva verzija *Visual Objects*-a obećava moćnu alatku za razvoj *Windows* aplikacija. S obzirom na kompatibilnost sa *Clipper*-om i broj razvijenih *Clipper* programa, VO može kod nas biti posebno interesantan.

Microsoft Access

PC Preporuka

Microsoft Access je poslednji stigao "u arenu" baza podataka - dBASE i njegovi naslednici suvereno su vladali tržištem već nekih petnaestak godina kada je *Microsoft* odigrao svoj adut. Činjenica da je već prva verzija *Access*-a izazvala ogromno interesovanje i prodata u stotinama hiljada primeraka svedoči o vanserijskom proizvodu. Verzija 2.0 donela je kompletnije okruženje, pouzdaniji rad i, uopšte, zrelost neophodnu za profesionalne primene.

Access je uspeo zato što je doneo nešto suštinski novo: umesto već toliko puta viđenih dBASE-ovskih ekrana i unosa podataka u @ ... GET polja, pojavili su se raskošni *Windows* ekrani, tabelarno prikazani podaci, *list* i *combo box*-ovi i, uopšte, stvari o kojima su *Clipper* programeri mogli samo da sanjaju. A sve se postizalo bez prevelikog truda, aktiviranjem nekog od "dežurnih čarobnjaka" (*Form Wizard* za kreiranje maski, *Report Wizard* za pripremu izveštaja itd).

Svo to "šarenilo" neminovno privlači pažnju, ali su prave vrednosti *Access*-a vidljive tek kada se pomnije prouči njegov podsistem za rad sa bazama podataka. Relacioni model je sinuo u punoj snazi: promovisan je princip primarnog ključa, automatizovana provera njegove jedinstvenosti, uvedene (automatske) kaskadne operacije koje osiguravaju ažurnost podataka u tabelama koje su povezane relacijama, realizovane transakcije koje omogućavaju da se u slučaju greške poništi čitava polovično obavljena operacija, automatizovan rad u mreži, obezbeđen link prema klijent / server arhitekturama...

Jedna od prednosti *Access*-a nad nekim drugim paketima koje prikazujemo je činjenica da ne morate programirati da biste radili sa njim. Naravno, određeni "programerski rezon" je neophodan u nekim fazama rada, naročito kada se postavljaju složeniji upiti. I tu je, ipak, mnogo toga automatizovano: pretraživanje po indeksima se, recimo, sa korisničkog aspekta uopšte ne razlikuje od pretraživanja po ostalim poljima: sve što treba da uradite je da u indeksiranim poljima aktivirate *Match Whole Field* opciju i ubrzanje dolazi samo od sebe. Kada je potrebna nešto složenija pretraga, prelazi se na filtre: u pitanju su klasični selekcioni kriterijumi koji se snimaju na disk radi kasnijeg korišćenja. Od snimljenog filtra do pravog upita (*query*) je veoma mali: možete čak da aktivirate opciju *Save Filter as Query* i tako složeniji filtar pretvoriti u upit.

Access-ovi upiti predstavljaju svet za sebe. Samo ime *query* (upit) je znatno preraslo svoj originalni smisao: i dalje se, naravno, najčešće koristi tzv. *select query* koji izdvaja neke slogove prema zadatim uslovima, ali postoje i *union query*, *action query* (ujedinjavanje više tabela), *crosstab query* (proizvodi izveštaje u obliku radnih tabela), *pass-through* i *data definition query* (veza sa SQL bazama podataka)... U praksi ćete, uz *select query*, najčešće koristiti *action query* koji omogućava promenu vrednosti nekih polja; važna novost verzije 2.0 je što se pre stvarne promene nove vrednosti mogu pregledati.

Pri formiranju upita od velike je pomoći *Query Wizard*, ali njegove "čarolije" povremeno treba malo neutralisati: pokazuje se jednostavnijim, čak i za laika, da tu i tamo otkuca neki logički izraz, nego da kroz silne dodatne maske određuje koji se uslovi kombinuju operatorom "i", koji operatorom "ili", šta treba izdvojiti iz tabele...

Logičan korak koji sledi iza upita je programski jezik, *Access Basic*. Jezik je, kao što mu ime i govori, relativno sličan *Visual Basic*-u, naravno uz brojne ekstenzije koje omogućavaju efikasniji rad sa podacima. Dosledno je objektno orijentisan: sve operacije sa podacima se svode na primenu unapred definisanih metoda. U najjednostavnijem slučaju, metod može da bude i obično ime polja u tabeli - ukoliko, na primer, želite da prenesete vrednost polja 'naziv' u lokalnu promenljivu 'ime1', koristićete naredbu nalik na ime1 = tabela.naziv. Za razliku od klasičnih jezika, "međutačka" ovde nije isključivo separator hijerarhija u slogu: njome se referenciraju i programi, odnosno akcije. U *Clipper*-u biste, na primer, najpre definisali radni indeks komandom SET ORDER TO 3 a onda napisali seek ("Marija") i tako pronašli slog čije indeksno polje ima vrednost 'Marija'. U *Access*-u pišete baza.index = "Ime" a onda baza.Seek "=" "Marija" i obavljate istu funkciju.

Programiranje u grafičkom okruženju kao što je *Windows* ne može se svesti na klasično proceduralno razmišljanje - bilo bi previše složeno predviđati svaku akciju ili sled akcija korisnika kome se miš stavi u šake! Zato *Access* (kao i dBASE) sa okruženjem komunicira preko događaja (*event driven*). Jedan od događaja, na primer, može da bude klik na 'OK' taster neke od maski - kada korisnik to uradi, automatski će biti pokrenuta odgovarajuća procedura. To praktično znači da ne postoji nikakav "glavni program" (glavna procedura) koji bi preuzeo kontrolu i sve vreme upravljao sistemom. Sve je prepušteno korisniku, čije akcije 'OK', 'Cancel', 'Help', 'Resize' i sve druge samo pokreću procedure koje obavljaju željeni posao. Uz sasvim malo prakse, dolazi se do *Windows* aplikacije koja korisniku nudi veliku fleksibilnost, a za programera je razmerno jednostavna i laka za debagovanje. Mada moramo reći da pri tom debagovanju alati ugrađeni u sam *Access* pružaju pomoć mnogo manju nego što bi se očekivalo, pre svega zato što je debager, premda korektan, po koncepciji prilično zastareo.

Kada jednom napišete *Access* program, možda ćete poželeti da ga prodate nekome ko se bavi sličnim poslom. U slučaju osnovnog paketa *Access 2.0*, to će biti moguće samo ako i kupac ima kopiju *Access*-a; za pravu distribuciju morate da dokupite i *Access 2.0 Distribution Kit*. Glavna komponenta ovoga dodatka je *Setup Wizard*, program koji, na osnovu vaših odgovora na postavljena pitanja, kreira distribucione diskete sa programom. Distribucione diskete po svemu liče na opremu samog *Access*-a - na njima je veoma profesionalan SETUP koji propita korisnika za podatke, prekopira potrebne datoteke na njegov disk i na kraju eventualno startuje program koji demonstrira aplikaciju ili ispisuje README fajl. Sve sitne usluge, kao što je kreiranje ikona i njihovo uključivanje u neku od grupa *Program Manager*-a, takođe spadaju u "čaroliju" istalacionog programa koji je, gotovo bez vaše pomoći, *Setup Wizard* napravio.

Prilikom kreiranja instalacione verzije vaše aplikacije treba da donesete samo jednu važnu odluku: da li je prenosite na sistem na kome je *Access* instaliran ili ne. U prvom slučaju, mogu se preneti samo bitne komponente paketa. Za pravu prodaju programa svakako treba generisati punu *run time* verziju koja obuhvata deo paketa *Access* neophodan da se programi izvršavaju, ali bez mogućnosti njihovog daljeg razvoja. Na disk korisnika se instalira baza podataka, sve potrebne maske, makroi i (prevedeni) programi koje će on potpuno slobodno koristiti, na jednom ili više računara (ništa od zabrane kopiranja), ali koje neće moći potpuno slobodno da izvršava (Ctrl Break ne omogućava prekid programa koji radi u okviru *Run Time* modula, dok u samom *Access*-u ova poznata DOS funkcija savršeno deluje) niti da, što je posebno značajno, analizira njegov tok ili menja kod... Bar dok neko ne napiše neku vrstu *Access* "dekompajlera" ili disasemblera, kao što se dešavalo kod *Clipper*-a.

Svi ovi noviteti plaćeni su glomaznošću - *Access* normalno radi tek na 486 sistemima sa 8 megabajta memorije, a ni 16 me-

gabajta mu neće škoditi za neke složenije operacije koje obuhvataju istovremeni rad nekoliko komponenti *Office*-a. Ne štedi se ni na distribucionim disketama - kratak *Hello world* program se, po prevođenju, upisuje na četiri (!) diskete po 1.44 M. Sada znate zašto su svi *Windows* programi "-onoliki".

Ukoliko imate dobar računar i ne smeta vam da krenete iz početka sa učenjem programiranja, *Access* je prava stvar za vas. Ukoliko, sa druge strane, zaključite da vam je učenja bilo dosta, verovatno ćete se opredeliti za neki drugi RDBMS.

Microsoft FoxPro for Windows

FoxPro je "sledeća generacija" prvog velikog dBASE III klona koji se zvao *Fox Base* - baza podataka koja je, što se jezika i upotrebe tiče, bila sasvim slična dBASE-u, ali je donela i brojne ekstenzije. Što je najvažnije, donela je performanse: lisica na kutiji *Fox Base*-a nije bila simbol lukavstva, nego brzine! Posle nekoliko veoma uspešnih izdanja *Fox Base*-a (i *Fox Base Plus*), pojavila se daleko kompletnija verzija koja je dobila ime *FoxPro*, zatim se pojavio *FoxPro 2.0* koji je doneo brojna unapređenja i ubrzanja uključujući i po mnogo čemu revolucionarnu *Rushmore* tehnologiju. To je bilo sasvim dovoljno da se *Microsoft* zainteresuje za ovaj proizvod, otkupi firmu *Fox Software* i, tokom prošle godine, promoviše najpre verziju 2.5 a zatim i verziju 2.6 za DOS i *Windows*.

Iako se verzije za DOS i *Windows* po spoljnom izgledu veoma razlikuju, u suštini se radi o istom proizvodu: u paketu dobijate kompajler za generisanje tradicionalnih 16-bitnih i profesionalnih 32-bitnih aplikacija, zajedno sa bibliotekama za povezivanje sa drugim jezicima i drugim (pre svega SQL) okruženjima (*Client - Server Conectivity Kit* - kod nekih prodavaca se posebno naplaćuje). Ukoliko želite da koristite *FoxPro* za razvoj RDBMS programa koji bi išli na tržište, moraćete da dokupite i *Distribution Kit* koji omogućava pokretanje *FoxPro* aplikacija tamo gde *FoxPro* nije instaliran.

Verzija 2.6 je, inače, sasvim slična verziji 2.0: nova "kozmetika" uključuje niz *Wizard* programa koji olakšavaju i ubrzavaju pojedine operacije i približavaju paket početnicima (zanimljivo je da se *Wizard*-i uopšte ne pominju u dokumentaciji, moraćete da tražite kroz *help)*, a *Catalog Manager* olakšava integraciju pojedinih segmenata projekta na kome se radi. U funkcionalnom delu paketa, međutim, nema nikakvih izmena, a samim tim su i performanse ostale neizmenjene... ali i dalje izuzetno konkurentne!

Čemu *FoxPro* duguje svoju brzinu? Pre svega dobro organizovanom radu sa indeksima - umesto da svaki od njih bude u posebnom fajlu, svi indeksi koji karakterišu jednu tabelu koncentrisani su u zajednički .CDX fajl koji je, obzirom na relativno male dimenzije, većim delom u memoriji, što žestoko ubrzava pretraživanje baze. Rad sa indeksima je dalje ubrzan tehnologijom čudnoga imena, *Rushmore* - ime je, kažu, nastalo u spomen poznatog Hičkokovog filma "Sever - Severozapad" ("*North by Northwest*") koji je autor gledao kada mu je ova ideja pala na pamet. Tehnologiju ćemo objasniti na jednostavnom primeru: korisnik zahteva da se iz tabele izdvoje svi slogovi koji su fakturisani posle 1. marta 1995. godine, i to na iznos veći od 1,000 dinara. U klasičnom dBASE/*Clipper* okruženju to bi se obavilo komandom nalik na LOCATE FOR datum>CTOD("01/03/95") .AND. iznos>1000 i izvršilo tako što bi bio najpre pročitan prvi slog, zatim drugi, treći... sve dok se ne pronađe neki koji zadovoljava uslove ili ne stigne do kraja datoteke. Ako datoteka ima više hiljada ili više desetina hiljada slogova, ova operacija će trajati veoma dugo. No, zamislimo sada da je datoteka indeksirana po datumu fakturisanja - *Clipper* će pomenuti LOCATE izvršavati na potpuno isti način, ne obazirući se na ovaj indeks, ali će *FoxPro* koristiti indeks da pronađe fakture poslate posle 1. marta (što se obavlja praktično trenutno)

pa onda **samo kod njih** proveravati iznose i izdvajati veće od 1000. Umesto, dakle, da se pregleda cela baza podataka, pregledaju se samo oni slogovi koji zadovoljavaju uslove koji se mogu odrediti preko definisanih indeksa - ukoliko pažljivo isplanirate indekse, sva pretraživanja će zbilja biti drastično ubrzana!

Ubrzanje, ma koliko to jeretički zvučalo, potiče i otuda što je svaka tabela izdvojena u svoj DBF fajl - koliko god da je DBF format neracionalan, pa i pregažen vremenom, njegova jasna i čvrsta struktura primetno ubrzava razne operacije. Naravno, uz plaćanje određene cene: obzirom da je svaka tabela svet za sebe, nema centralne definicije relacija pa samim tim ni pomisli o nekoj automatskoj proveri konzistentnosti podataka. Na programeru je da brine o tome da ažuriranje jedne tabele istovremeno ažurira i prateće; pri tome se mora ozbiljno paziti na pojavu greški, pošto *FoxPro* ne podržava transakcije kojima bi se neka neuspela kaskadna operacija mogla poništiti. Mana je i nepostojanje primarnog ključa, a ima i stvari koje DBF naprosto ne podržava a trebalo bi ih podržati. Tu pre svega mislimo na tip *currency* (zgodan za rad sa novčanim iznosima) i OLE objekte. Jedina velika prednost *FoxPro* formata nad nekadašnjim dBASE-om su neograničena *memo* polja proizvoljnog sadržaja. U njih se, dakle, mogu upisivati tabele, slike i drugi objekti ali to ne obezbeđuje OLE veze sa drugim *Windows* aplikacijama - sadržaj je potpuno statičan.

FoxPro je posebno razvijen u tri segmenta: SQL, biblioteka funkcija i rad sa projektima. Kompletan SQL "krije" se iza komande SELECT: željeni podaci izdvajaju se iz baze navođenjem proizvoljno složenih uslova. Početnici će se verovatno opredeliti za izdvajanje bez kucanja naredbi *FoxPro* programskog jezika, tj. raditi sa *query by example* mehanizmima koji omogućavaju zadavanje upita klikom na postojeće podatke i navođenjem graničnih vrednosti. Nevolja je jedino u tome što SELECT komanda nije baš potpuno dovršena: izdvojeni skup slogova se može proizvoljno pregledati, ali se u njega ne mogu unositi izmene. Bilo koji *subset* je, efektivno, *read-only*.

Fox je od samih početaka jak po pitanju rada na većim projektima: na osnovu fajla sa ekstenzijom PJX zna se koji svi moduli, definicije baza, ekrana i svega ostalog čine tekući projekat, pa svaka izmena, prevođenje, arhiviranje i druga operacija na nivou projekta predstavlja jedan klik mišem na neku od *Build Project* opcija. Razni projekti mogu da dele iste baze podataka, potprograme, ekrane, formate izveštaja... važno je samo da se u okviru posebnog niza maski precizno zada lokacija svake komponente. Najsloženiji podsistem "editora projekta"

je, naravno, segment za dizajniranje ekrana, koji je u verziji 2.6 rešen u dva nivoa. Na najjednostavnijem nivou, *Screen Wizard* omogućava potpuno automatsko formiranje maske, ali uz jedno vrlo ozbiljno ograničenje: u formularu mogu da figurišu samo polja iz jedne baze! Ukoliko, što je u praksi mnogo češći slučaj, treba istovremeno da radite na podacima iz više tabela, koristićete *Screen Builder* i njegove alate: nije jednostavno za upotrebu kao *Screen Wizard*, ali uz malo prakse sve brzo dolazi na svoje mesto.

FoxPro je pravi šampion što se tiče ugrađenih funkcija i procedura - lista je toliko široka i detaljna da ponekad prelazi u svoju suprotnost, tj. dovodi do problema u pronalaženju željene funkcije u pravom moru sličnih. Naročito je kompletan set funkcija za rad sa alfanumericima (postoji i fonetsko poređenje, doduše prilagođeno engleskom jeziku), nizovima i matricama - ako *FoxPro* nema funkciju za ono što vam je potrebno, možemo samo reći da imate veoma neobične potrebe!

Posmatran kao softverski paket, *FoxPro* se mora oceniti najvišom ocenom, pre svega zbog izuzetnih performansi - onom ko je radio sa *Fox*-om, nikada neće biti jasno zašto ostali moraju da budu tako spori. Posmatran kao kompletan proizvod, dakle zajedno sa strategijom razvoja i marketinga, *FoxPro* pokazuje veoma ozbiljne mane, koje se mogu svesti na tvrdnju da se sa njegovim razvojem stalo kada ga je *Microsoft* kupio. Tu pre svega mislimo na veze *FoxPro*-a i drugih *Windows* aplikacija: dok konkurentski paketi, dopunjeni odgovarajućim drajverima, bez ikakvih problema pristupaju raznim tipovima podataka, *FoxPro* je ograničen na DBF fajlove - sve drugo se mora *import*-ovati tj. prevesti u DBF. Dok drugi paketi sa lakoćom komuniciraju preko OLE objekata, *FoxPro* je tek osposobljen da bude OLE 1.0 klijent, ali se njegove baze podataka ne mogu ugrađivati u druge *Windows* aplikacije (OLE server). *Unix* verzija *FoxPro*-a, koja je nekada bila vrlo jak adut za *Fox* (korišćenje istog alata na raznim platformama) i dalje je "u razvoju". Sve u svemu, izgleda da se *Microsoft* još nije odlučio šta će raditi sa *Fox*-om: da li će nastaviti da ga razvija kao poseban proizvod, ili će ono što je u njemu najvrednije (pre svega *Rushmore* tehnologiju) ugraditi u *Access* i jednostavno "ugasiti" *FoxPro*. Za one koji tek počinju da rade sa bazama podataka, ova neizvesnost je sasvim dovoljna da *FoxPro* ne uzimaju u obzir; za one koji ga već sa uspehom koriste, on ostaje izuzetno upotrebljiv alat za proizvodnju profesionalnih DBMS aplikacija!

Powersoft Power Builder

Dejan Vesić

Power Builder firme *Powersoft* kod nas (još?) nije naročito rasprostranjen, ali na kanadskom i američkom tržištu predstavlja prilično široko zastupljenu alatku za rad sa bazama podataka pod *Windows*-om. Na tržištu se nalaze dve osnovne verzije paketa: *Power Builder Desktop* i *Power Builder Enterprise*. *Desktop* je slabija varijanta, dok je *Enterprise* ekvivalent profesionalnih izdanja sličnih paketa.

Najnovija verzija Desktop-a (3.0a) dolazi na 8 disketa po 1.44 megabajta. Po instalaciji zauzima skromnih (?) 16 megabajta na disku. Ovo je jedna od retko "kulturnih" aplikacija po pitanju instalacije - sve svoje DLL fajlove stavlja u svoj direktorijum, zaobilazeći ionako prepun C:\WIN\SYSTEM. Hardverski zahtevi nisu preveliki: radiće (i to sasvim pristojno) na 386DX sa 4 M memorije mada mu, prirodno, jače mašine sa više RAM-a neće ni malo "škoditi". Programi koje generiše PB će raditi i na slabijim mašinama (386SX/2Mb); kao ilustracija, čuveni "Hello World" se kompajlira u HELLO.EXE od 9 kilobajta (*Access* će ga napraviti na 4-5 disketa). Oduševljenje se ponešto smanjuje kada se sazna da uz generisani EXE fajl treba isporučiti i standardne DLL-ove iz paketa *Power Builder*.

Kao sloj prema konkretnom fizičkom zapisu podataka PB ima na raspolaganju čitavu lepezu drajvera: počev od standardnog i pomalo prevaziđenog DBF formata (dBASE III i IV, *Clipper* 5.x), preko ASCII TXT fajla (!), *Access 1.x* i *Btrieve 5.x* formata do *Excel* i *FoxPro* (2.x za DOS i *Windows*), *Paradox* (3.x i 4.x za DOS) i neizbežnog SQL-a (WATCOM i *NetWare* 3.x SQL). Korisnici profesionalne verzije na raspolaganju imaju i egzotiku kao što je pristup DB2 i *Informix* bazama podataka. Prirodno okruženje PB-a je SQL; uz *Desktop* stiže WATCOM SQL 3.2b demo (demo, jer niste u mogućnosti da kreirate novu bazu podataka već se sve vaše tabele smeštaju u PBDEMODB.DB; takođe se radi o jednokorisničkoj varijanti).

PB je objektno orijentisani sistem. Hijerarhijski najviši objekat je aplikacija; sledе *DataBase, DataWindow, Window, MDI Client* (PB standardno omogućava izradu MDI (*Multiply Document Interface*) aplikacija) i *User Object* (za specifične zahteve korisnika - programera, koje nije moguće izvesti kroz standardne objekte). Izrada aplikacije teče kroz *Painter*-e, po sistemu *Drag & Drop*. U zavisnosti od toga koji objekat "crtate" aktivira se odgovarajući tasterski meni sa alatima koje imate na raspolaganju.

Izrada jedne aplikacije teče u sedam faza. U prvoj kreirate potreban projekat, dok druga, znatno obimnija, obuhvata definisanje (jednog ili više) *DataBase* objekata, opisa tabela u kojima se nalaze vaši podaci. Neophodno je kreirati i negrafički transakcioni objekat zadužen za stvarnu komunikaciju sa podacima na disku. Preko atributa transakcionog objekta se povezuje baza podataka sa aplikacijom, zahteva operacija nad podacima i saznaju rezultati zahtevane operacije. PB postavlja kao podrazumevani transakcioni objekat SQLCA.

U trećoj fazi kreirate *DataWindow* objekat, koji se najčešće vezuje za *DataBase* - zadužen je za izdvajanje podskupa podataka iz tabela (SELECT iskaz SQL-a). Sledi četvrta faza u kojoj se kreiraju *Window* objekti (prozori), sloj između korisnika i aplikacije. Sve standardne kontrole (*SingleLineEdit, MultiLineEdit, Choice, Buttons*...) su na raspolaganju.

Za svaku kontrolu/objekat i poruku treba, u petoj fazi, napisati skript u *PowerScript* jeziku koji prilično podseća na bejzik. Poznaje sve standardne naredbe: IF / ELSE / ENDIF, DO CASE i sve kombinacije DO LOOP (While, Until). I tipovi podataka su standardni: *Boolean, Char, Date, DateTime, TimeStamp* (posebno pogodan kao primaran ključ u tabelama), *Integer, Double, String*) i manje standardni: Blob (*Binary Large OBject* - moguće smeštati sve i svašta, uključujući i slike), *Object, PowerObject* i *DragObject*. Ostaje da u finalnoj fazi testirate vašu aplikaciju, uz korišćenje sasvim solidnog debagera. Kada je sve isprobano, generišete i distribuirate finalni EXE fajl.

Power Builder je zaokružen proizvod uz pomoć koga možete brzo (ali stvarno **brzo**) generisati aplikaciju profesionalnog izgleda i preformansi. Ima i svojih mana: čudno organizovan help, mnogo manje "šminke" od konkurentskih proizvoda, nedostatak raznih *Wizard* alata (nema čak ni *Setup Wizard*-a)... ali je zato daleko manje zahtevan i ima izuzetan izbor formata kojima može pristupit. Za onog ko prelazi sa DOS platforme i *Clipper* okruženja, prelazak u "prozore" neće biti bezbolan. Ali, ako imate iole iskustva u objektnom i SQL programiranju, možda ćete se i sami iznenaditi kako je sve prirodno i lako. U svakom slučaju, ovu alatku vredi probati i videti na delu.

	dBASE	Paradox	Power Builder	Visual Objects	Access	FoxPro
Verzija	**5.0**	**5.0**	**3.0a**	**1.0**	**2.0**	**2.6**
Okruženje	*DOS,Win*	*DOS,Win*	*Win*	*Win*	*Win*	*DOS,Win,Mac*
Zahtevi	386 4M	386 4-8M	386 4M	386 4-8M	386 4-8M	386 4M
Ograničenja						
Max. polja	1024	255	255	65534	255	256
Max. kolona	255	255	999	> 1000	255	255
Max. slogova	2 milijarda	1 milijarda		1 milijarda	1 milijarda	1 milijarda
Max. indeksa	255	255	32767	> 1000	32	255
Tipovi podataka						
Celi brojevi	Da	Da	Da	Da	Da	Ne
Racionalni, fiksni zarez	Da	Ne	Da	Ne	Da	Da
Racionalni, pokretni zarez	Da	Da	Da	Da	Da	Ne
Datum	Da	Da	Da	Da	Da	Da
Bulovi	Da	Ne	Da	Da	Da	Da
Memo	Beskonačno	Beskonačno	60 K	Beskonačno	64 K	Beskonačno
Binarni objekti	Da	Da	Da		Da	Da
Stringovi (fiksni)	Da	Da	Da	Da	Ne	Da
Stringovi (varijabilni)	Ne	Da	Da	Da	Da	Ne
Integritet podataka						
Primarni ključ	Ne	Da	Da	Ne	Da	Ne
Provera jedinst. vrednosti	Ne	Ne	Da	Ne	Da	Ne
Kaskadna ispravka	Ne	Da	Da	Ne	Da	Ne
Kaskadno brisanje	Ne	Ne	Da	Ne	Da	Ne
Query						
xBASE kompatibilan	Da	Ne	Ne	Da	Ne	Da
Zadavanje u comm liniji	Da	Ne	Da	Ne	Da	Da
Zadavanje po primeru	Da	Da	Da	Ne	Da	Da
Prevođenje i optim. upita	Ne	Ne	Ne	Da	Da	Da
Max. polja za pretragu	Beskonačno	255	Beskonačno	Beskonačno	256	Beskonačno
ODBC kompatibilan	Da	Ne	Da	Da	Da	Da
Rad u mreži						
Zaključavanje baze	Da	Da	Da	Da	Da	Da
Zaključavanje sloga	Da	Da	Da	Da	Da	Da
Zaključavanje polja	Ne	Ne	Ne	Ne	Ne	Ne
Automat. timeout kontrola	Da	Da	Da	Da	Da	Da
Rad sa transakcijama	Da	Ne	Ne	Ne	Da	Ne
Razvojne alatke						
Generator menija	Da	Ne	Da	Da	Da	Da
Generator izveštaja	Da	Da	Da	Da	Da	Da
Runtime verzija	Ne	Da	Ne	-	Da	Da
Samostalni EXE	Ne	Ne	Da	Da	Ne	Ne