FAKULTET ZA POSLOVNU INFORMATIKU

BAZA PODATAKA U KARATE KLUBU

- Diplomski rad -

Mentor:
Prof dr Mladen Vejnović

Student:
Branko Graovac

Br. indeksa:
6S/2003

Univerzitet Singidunum
Beograd, 2007.

FAKULTET ZA POSLOVNU INFORMATIKU

UNIVERZITET SINGIDUNUM
FAKULTET ZA POSLOVNU INFORMATIKU
Beograd, Danijelova 32

Broj: _____/2007

Kandidat: Branko Graovac

Broj indeksa: 6S/2003

Smer: Projektovanje i programiranje

Tema: BAZA PODATAKA U KARATE KLUBU

Zadatak: Realizovati i opisati bazu podataka u karate klubu. Unos takmičara, rezultata i ostalih evidencija u klubu u bazu podataka. Omogućiti prikaz, izmene, brisanje i štampanje iz baze. Za rešenje zadatka primeniti programski jezik Microsoft Access 2003.

MENTOR

_____
Prof. dr Mladen Veinović

Datum odobrenja teme:
03.06.2007.
Beograd

DEKAN

_____
Prof. dr Milan Milosavljević

SAŽETAK

Mnogo je korisnika Microsoft Officea. Ipak, većina ih je navikla da svoje potrebe za sređivanjem dokumentacije zadovoljiti kroz uzak deo spektra mogućnosti ovoga kancelarijskog paketa. MS Office nas uvodi u svet baza podataka, uz pomoć popularnog i moćnog programa Microsoft Accessa.

MS Access je jedan od programa koji su sastavni deo paketa programa MS Office. Access koristimo za kreiranje i upravljanje bazama podataka. Ako bi želeli da definišemo bazu podataka možemo reći da je to kolekcija, velika količina podataka sumirana na jednom mestu, a odnosi se na neki subjekat, pojavu, namenu, kao što je vođenje magacina, poslovanje video kluba, biblioteke, sportskog kluba i sl.

U radu koji je pred nama objasniću osnovne mogućnosti i tehnike Accessa u kreiranju baza podataka i upravljanja podacima koji se u njima nalaze. Da bi ovaj postupak bio lakši, od samog početka radićemo na primeru baze podataka neophodne za rad karate kluba „ENPI“ iz Subotice, a koja obuhvata evidenciju članova kluba, takmićenja, postignutih rezultata, izveštaje i sl. Po završetku imaćemo gotovu aplikaciju za baratanje podacima, koju smo sami izradili uz pomoć MS Accessa.

ABSTRACT

There are many users of Microsof Office. Still, when dealing with their documents, many of them are satisfied with using just a small section of the wide range of possibilities offered by this office package. MS Office leads us into the world of databases with the popular and powerful Microsoft Access programme.

This is one of the programmes included in the MS office program package. Access is used for creating and managing databases. If we wanted to define a database, we could say that it is a collection or a large amount of data accumulated in one place, covering a subject, occasion or purpose such as managing a warehouse, running the business of a club, library, sports club etc.

In this paper I intend to present the basic possibilities and techniques used by Access in creating databases and managing the data contained in them. To facilitate the tash, we shall use a practical example of a database necessary to run the ENPI karate club from Subotica, which includes records of team members, competitions, results achieved, reports etc. At the outcome, we shall have a finished database for handling data, made with MS Access.

SADRŽAJ

1 UVOD ..... - 1 -
1.1 Šta je to baza podataka? ..... - 1 -
2 ACCESS KAO ALAT ZA PROGRAMIRANJE ..... - 2 -
2.1 Kakve tipove aplikacija možemo napraviti u Accessu? ..... - 2 -
2.1.1 Access kao razvojna platforma za personalne aplikacije ..... - 2 -
2.1.2 Access kao razvojna platforma za poslovne aplikacije malih firmi ..... - 2 -
2.1.3 Access kao razvojna platforma za aplikacije odeljenja ..... - 3 -
2.1.4 Access kao razvojna platforma za aplikacije korporacije ..... - 3 -
2.1.5 Access kao front end za klijent/server aplikacije preduzeća ..... - 4 -
2.1.6 Access kao razvojna platforma za aplikacije u lokalnoj mreži/lnternetu . - 5 -
2.2 Access kao proizvod promenljive veličine ..... - 5 -
3 ŠTA JE SVE NOVO U ACCESSU 2003? ..... - 6 -
3.1 Nove osobine formulara ..... - 6 -
3.2 Nove osobine izveštaja ..... - 7 -
3.3 Uzbudljivi svet pogleda PivotTable i PivotChart ..... - 7 -
3.4 Access 2003 i XML ..... - 7 -
3.5 Nove osobine ADP projekata ..... - 8 -
3.6 Nove osobine stranica za rad sa podacima ..... - 8 -
4 KAKO DA POČNEM DA PRAVIM ACCESSOVU APLIKACIJU? ..... - 9 -
4.1 Analiza zadatka ..... - 9 -
4.2 Analiza i struktura podataka ..... - 9 -
4.3 Testiranje ..... - 10 -
4.4 Pravila imenovanja objekata ..... - 10 -
5 UPOZNAJMO SE SA ACCESS - om ..... - 11 -
6 OSNOVNI ELEMENTI ACCESS - a ..... - 12 -
7 OBJEKTI BAZE ..... - 14 -
8 TABELE: Skladište podataka ..... - 14 -
8.1 Odabiranje odgovarajućeg tipa polja za podatke ..... - 15 -
8.1.1 Polja tipa Text: Najčešći tip polja ..... - 16 -
8.1.2 Polja tipa Memo: Za velike beleške i komentare ..... - 17 -
8.1.3 Polja tipa Number: Za slučajeve kada treba obaviti izračunavanja ..... - 17 -
8.1.4 Polja tipa Date/Time: Za evidentiranje događaja ..... - 17 -
8.1.5 Polja tipa Fields: Zapisivanje novčanih vrednosti ..... - 17 -
8.1.6 Polja tipa AutoNumber: Za jedinstvene identifikatore slogova ..... - 18 -
8.1.7 Polja tipa Yes/No: Kada je jedan od dva odgovora tačan ..... - 18 -
8.1.8 Polja tipa OLE Object: Polje u koje možemo smestiti gotovo bilo šta .. - 18 -
8.1.9 Polja tipa Hyperlink: Vaša veza sa Internetom ..... - 19 -
8.1.10 Svojstvo Field Size: Kako ograničiti unošenje u polje ..... - 19 -
8.1.11 Svojstvo Format: Način prikazivanja podataka ..... - 19 -
8.1.12 Svojstvo Input Mask: Kakvi podaci se smeštaju u polje ..... - 19 -

8.1.13 Svojstvo Caption: Velika ušteda vremena ... - 20 -
8.1.14 Svojstvo Default Value: Uštedite vreme prilikom unošenja podataka ... - 21 -
8.1.15 Svojstvo Validation Rule: Kontroliše šta korisnik unosi u polje ... - 21 -
8.1.16 Svojstvo Validation Text: Kako obezbediti poruku za korisnika ... - 21 -
8.1.17 Svojstvo Required: Naterajte korisnika da unese vrednost ... - 21 -
8.1.18 Svojstvo Allow Zero Length: Prilagodite polje za nepostojeće podatke - 22 -
8.1.19 Svojstvo Indexed: Ubrzajte pronalaženje podataka ... - 22 -
8.1.20 Svojstvo Unicode Compression: Komprimovanje podataka ... - 23 -
8.1.21 Korišćenje osobine Lookup ... - 23 -
8.2 Rad sa svojstvima tabele ... - 24 -
8.2.1 Korišćenje indeksa radi poboljšanja performansi ... - 25 -
8.2.2 Accessove tabele i Internet ... - 25 -
8.2.3 Polje tipa Hyperlink ... - 25 -
8.3 Ključ integriteta podataka ... - 26 -
8.4 Ciljevi strukturiranja relacione baze podataka ... - 26 -
8.5 Pravila strukturiranja relacione baze podataka ... - 26 -
8.6 Tipovi zavisnosti između tabela ... - 26 -
8.6.1 Zavisnost jedan-prema-više ... - 27 -
8.6.2 Zavisnost jedan-prema-jedan ... - 27 -
8.6.3 Zavisnost više-prema-više ... - 28 -
8.7 Uspostavljenje zavisnosti u Accessu ... - 28 -
8.8 Uspostavljanje zavisnosti između dve tabele ... - 29 -
8.8.1 Menjanje postojećih zavisnosti ... - 29 -
8.8.2 Uspostavljanje referencijalnog integriteta ... - 30 -
8.8.3 Kaskadno ažuriranje povezanih polja ... - 31 -
8.8.4 Kaskadno uklanjanje povezanih slogova ... - 32 -
8.8.5 Dobrobiti zavisnosti ... - 33 -
8.8.6 Proučavanje indeksa i zavisnosti ... - 33 -
8.9 Istorijat relacionih baza podataka ... - 33 -
8.10 Pravila za jedinstvenost i ključeve ... - 34 -
8.10.1 Pravila za strane ključeve i domene ... - 34 -
9 Upiti ... - 35 -
10 IZVEŠTAJI: Pretvaranje podataka u informacije ... - 38 -
11 Makroi i VBA ... - 39 -
11.1 Makroi: Način za automatizovanje sistema ... - 39 -
11.2 Moduli: Osnova postupka pravljenja aplikacije ... - 40 -
12 NORMALIZACIJA I NORMALNE FORME ... - 41 -
12.1 Prva normalna forma ... - 41 -
12.2 Druga normalna forma ... - 41 -
12.3 Treća normalna forma ... - 41 -
12.4 Denormalizacija - namerno kršenje pravila ... - 42 -
12.5 Pravila integriteta ... - 42 -
12.6 Opšta pravila ... - 42 -
13 MODELOVANJE PROCESA ... - 43 -
14 MODEL PROCESA – KARATE KLUBA ... - 45 -
14.1 KONTEKSTNI DIAGRAM «IS karate kluba» ... - 45 -
14.2 NULTI NIVO DEKOMPOZICIJE ... - 46 -
14.3 PRVI NIVO DEKOMPOZICIJE «obrada podataka o takmičaru» ... - 47 -

14.4 PRVI NIVO DEKOMPOZICIJE «takmičarska ekipa» ..... - 48 -
14.5 DRUGI NIVO DEKOMPOZICIJE «unos novog takmičara» ..... - 49 -
14.6 TREĆI NIVO DEKOMPOZICIJE «ostvareni rezultati» ..... - 50 -
14.7 ČETVRTI NIVO DEKOMPOZICIJE «sezona» ..... - 51 -
14.8 HIJERARHIJSKO STABLO PROCESA «IS karate kluba» ..... - 52 -
14.9 HIJERARHIJSKO STABLO PROCESA «obrada podataka o takmičaru» ..... - 53 -
14.10 HIJERARHIJSKO STABLO PROCESA «takmičarska ekipa» ..... - 54 -
15 MODELOVANJE PODATAKA ..... - 55 -
16 POKRETANJE PROGRAMA, PRIPREMNE RADNJE, KREIRANJE PRVE TABELE ..... - 56 -
16.1 Planiranje osnove ..... - 57 -
16.2 Planiranje tabela ..... - 58 -
16.3 Po istom principu će mo kreirati i ostale tabele ..... - 61 -
17 POVEZIVANJE TABELA ..... - 64 -
18 UNOŠENJE PRVIH ZAPISA ..... - 71 -
19 KREIRANJE ELEKTRONSKIH OBRAZACA - FORMI ..... - 73 -
19.1 Prva forma – Talmicari ..... - 73 -
19.2 Kontrole u formi ..... - 77 -
19.3 Događaji (eng. Events) ..... - 79 -
19.4 Makro program ..... - 80 -
19.5 VBA program ..... - 81 -
19.6 Pretraga u formi ..... - 82 -
19.7 Samostalno padajuća lista ..... - 85 -
19.8 Podforma (podmaska ili podobrazac) ..... - 85 -
19.9 Forma sa podformom ..... - 86 -
20 KOMANDNA TABLA ..... - 88 -
20.1 Izgled komandne table ..... - 88 -
21 FUNKCIJE KOMANDNE TABLE ..... - 91 -
21.1 Potvrda upisa sloga kod unosa na formi ..... - 94 -
21.2 Aktiviranje prekidača (ChekBox) na formi ..... - 95 -
22 QUERY ..... - 97 -
23 IZVEŠTAJI ..... - 102 -
23.1 Sprečimo pojavu praznog izveštaja ..... - 105 -
24 ZAVRŠNE RADNJE UREĐENJA PROGRAMA ..... - 106 -
25 POZDRAVNI EKRAN ..... - 111 -
26 KAKO NAPRAVITI ACCESS APLIKACIJU ..... - 112 -
27 PODELIMO MDB DATOTEKU NA PODATKE I APLIKACIJU ..... - 113 -
28 RAZLOZI ZA ODRŽAVANJE BAZE ..... - 114 -
29 RAD PROGRAMA SA DVE ILI VIŠE RADNIH STANICA MREŽE ..... - 115 -
30 PRAKTIČNI SAVETI ZA RAD U ACCESSU ..... - 116 -
31 ZALJUČAK ..... - 119 -
32 LITERATURA ..... - 120 -



1 UVOD

Dakle, šta je to mitsko biće, poznato pod imenom **relaciona baza podataka** (engl. relational database)? Ukratko, baza podatka je alatka koja omogućava skladištenje podataka i rad s njima, na efikasan i delotvoran način. „Efikasno i delotvorno" znači da su podaci zaštićeni od nenamernog gubljenja ili oštećenja, da se za tu namenu ne koristi više resursa (ljudskih ili računarskih) nego što je zaista neophodno i da se podaci mogu učitavati u smislenim oblicima unutar prihvatljivih ograničenja performansi.

Teorijski, softver za relacionu bazu podataka može se napisati "od nule", ali u praksi ćemo uvek koristiti usluge određenog sistema za upravljanje bazama podataka (*database management system*, DBMS). DBMS se ponekad naziva relacioni DBMS (RDBMS), ali tehnički gledano.

Access je sistem za upravljanje bazama podataka **(data base management system DBMS)**, koristi se za skladištenje i učitavanje podatke, prikazivanje, a takođe i za automatizaciju poslova koji se često ponavljaju. Predstavlja odličnu kombinacija produktivnosti programa za upravljanje bazom podataka i lakoće korišćenja Windowsa.

Kao zaokružena aplikacija, Access obezbeđuje potpuno upravljanje relacionom bazom podataka i integritet na nivou same mašine na kojoj se nalazi baza (što sprečava ne potpuno ažuriranje i brisanje). Tabelama je moguće pridružiti pravila koja proveravaju ispravnost i validnost podataka i sprečavaju unos neispravnih, a svako polje tabele može imati svoj format i podrazumevanu vrednost što znatno povećava produktivnost.

Jedan od nedostataka Accessa je količina hardveskih resursa koji su mu neophodni.

1.1 Šta je to baza podataka?

Terminologija baza podataka gotovo je isto toliko neodređena koliko i izraz „objektno orijentisano programiranje". Izraz „baza podataka" koristi se za opisivanje svačega, u opsegu od obične grupe podataka, do složenog skupa alatki, kao što je SQL Server, a i svega između. Taj manjak preciznosti ne mora obavezno da znači nešto loše - razlog je sama priroda jezika.

Baze podataka su svojevrsne organizovane "elektronske kartice", kolekcija informacija vezanih za određeni subjekt, namenu ili pojavu, kao što su praćenje narudžbina kupaca, sportskog kluba, vođenje stanja magacina ili održavanje muzičke kolekcije u prividnom tabelarnom izgledu.

Bazu podataka ne čine samo organizovane tabele kao nosioci podataka, već je to kompletan program koji može proračunavati, filtrirati podatke, štampati izveštaje itd. Programe sličnog tipa možemo najčešće sresti u preduzećima, bankama, poštama, arhivama, i praktično rečeno - svuda.



2 ACCESS KAO ALAT ZA PROGRAMIRANJE

2.1 Kakve tipove aplikacija možemo napraviti u Accessu?

Često se nalazim u situaciji da sebi objašnjavam kakvi tipove aplikacija se mogu napraviti pomoću Microsoft Accessa. U Accessu postoji mnoštvo osobina koje zadovoljavaju različite potrebe baze podataka. Možemo ih koristiti za pravljenje šest osnovnih tipova aplikacija:

- Personalne aplikacije
- Aplikacije za male firme
- Aplikacije za odeljenja firme
- Aplikacije korporacije
- Aplikacije koje su front end klijent/server aplikacija
- Aplikacije za Internet ili lokalnu mrežu

2.1.1 Access kao razvojna platforma za personalne aplikacije

Na najnižem nivou, Access možemo koristiti za pravljenje jednostavnih personalnih sistema za upravljanje bazama podataka. Ljudi koji Access kupuju kako bi automatizovali sve, od kolekcije vina do kućnih finansija, često su razočarani. Problem leži u tome što je Access varljivo lagan za korišćenje. Njegovi sjajni ugrađeni čarobnjaci stvaraju utisak da je Access proizvod koji se lako može koristiti. Pošto odgovorimo na niz pitanja, završili smo pravljenje razvodnih tabli aplikacije, ekrane za unos podataka, izveštaje i tabele u kojima se nalaze podaci. Iako jeste tačno da najjednostavnije Accessove aplikacije možemo napraviti, a da uopšte ne razmišljamo o strukturi i da ne napišemo niti jedan red koda, za većinu aplikacija je ipak neophodno da napišemo kod i da ih struktuiramo.

Sve dok smo zadovoljni personalnim aplikacijama koje generiše čarobnjak, uz male izmene, neće se javiti nikakvi problemi. Problemi se mogu javiti kada poželimo da značajno izmenimo personalnu aplikaciju.

2.1.2 Access kao razvojna platforma za poslovne aplikacije malih firmi

Access je odlična platforma za pravljenje aplikacija koje mogu koristiti male firme. Njegovi čarobanjaci omogućavaju programerima da brzo i lako naprave osnovu aplikacije. Mogućnost pravljenja modula koda omogućava programerima pravljenje biblioteka funkcija koje se mogu koristiti na više mesta, a mogućnost pravljenja koda formulara i izveštaja im omogućava da naprave moćne formulare i izveštaje.



2.1.3 Access kao razvojna platforma za aplikacije odeljenja

Accessove performanse su adekvatne za većinu aplikacija koje bi se koristile po odeljenjima bez potrebe za korišćenjem klijent/server tehnologije. Najzad, većina odeljenja velikih korporacija ima razvojne budžete koji omogućavaju pravljenje dobrih aplikacija. U mnogim odeljenjima postoji stručnjak za računare koji je voljan da pomogne u pravljenju formulara i izveštaja. To odeljenjima stvara utisak vlasništva jer su dali svoj doprinos pravljenju aplikacije. To takođe posao programera čini lakšim.

2.1.4 Access kao razvojna platforma za aplikacije korporacije

Iako je Access možda najpogodniji za pravljenje aplikacija koje će se koristiti u odeljenjima firme, možemo ga koristiti za pravljenje aplikacija koje će se koristiti u celoj firmi. Koliko je uloženi trud vredan, zavisi od korporacije. Postoji granica broja korisnika koji istovremeno (konkurentno) mogu da koriste Accessovu aplikaciju, a da pri tome ona zadrži prihvatljive performanse, a takođe postoji i ograničenje broja slogova neke tabele, a da se značajno ne degradiraju performanse. Ovi brojevi zavise od slededh faktora:

- Koliko mrežnog saobraćaja već postoji u mreži?
- Koliko RAM memorije i procesora postoji u serveru?
- Kako se server već koristi? Na primer, da li se aplikacije, kao što je Microsoft Office, učitavaju sa servera ili sa lokalnih radnih stanica?
- Kakve vrste poslova korisnici aplikacije obavljaju? Da li postavljaju upite, unose podatke, prave izveštaje i tako dalje?
- Gde se Access i Accessove aplikacije izvršavaju, na serveru ili radnoj stanici?
- Koji se mrežni operativni sistem koristi?

Opšte pravilo za Accessove aplikacije koje nisu klijent/server bazirane jeste da se loše performanse dobijaju kada ima više od 10-15 istovremenih korisnika i kada postoji više od 100.000 slogova.

Programeri često ne razumeju šta je zapravo Access, a šta nije, kada se posmatra klijent/server platforma baze podataka. Često se postavlja pitanje: "Nije li Access klijent/server baza podataka?" Odgovor je da je Access neobičan proizvod jer je to aplikacija servera datoteka, ali se može koristiti kao front end za klijent/server bazu podataka. Evo objašnjenja: ukoliko kupimo Access i napravimo aplikaciju koja podatke čuva u Accessovoj bazi podataka na serveru datoteka, onda se na radnim stanicama obavlja sva obrada podataka. To znači da svaki put kada korisnik zada upit ili zatraži izveštaj, server datoteka radnoj stanici šalje sve podatke. Radna stanica zatim izvršava upit, a rezultat prikazuje u tabelarnom prikazu ili izveštaju. Ovakav postupak generiše značajan mrežni saobraćaj, naročito ukoliko više korisnika istovremeno traži izveštaje ili zadaje upite nad velikim Accessovim tabelama. Zapravo, ovakve operacije celu mrežu mogu potpuno zagušiti.



2.1.5 Access kao front end za klijent/server aplikacije preduzeća

Klijent/server baza podataka, kakve su Microsoft SQL Server ili Oracle, upite obrađuje na serverskom računaru, a rezultate šalje radnoj stanici. Pošto serverski softver ne može korisniku prikazati podatke, tu u pomoć priskače Access. Ponašajući se kao front end, Access može u izveštajima, tabelama ili formularima prikazati podatke dobijene sa servera baza podataka. Ukoliko korisnik ažurira podatke u Accessovom formularu, radna stanica šalje izmene back-end bazi podataka. Ovaj postupak možemo sprovesti ukoliko ostvarimo vezu sa spoljašnjim bazama podataka tako da nama i korisniku izgleda kao da radimo sa Accessovim tabelama, ili korišćenjem tehnike kojima se klijent/server podacima direktno pristupa.

Koristeći Accessove datoteke projekta, možemo napraviti aplikaciju koja je namenjena klijent/server okruženju. Ove datoteke projekta, koje su poznate kao Access Data Project (ADP) datoteke, zadržavaju formulare, izveštaje, makroe, module i stranice za rad sa podacima programa. Projekat je povezan sa back-end bazom podataka u kojoj se nalaze tabele, uskladištene procedure, pogledi i dijagrami baze podataka koje program koristi. Iz datoteke projekta lako možemo menjati i manipulisati objektima koji se nalaze na serveru, koristeći Access grafički korisnički interfejs. ADP datoteke pomažu da brzo pravljenje aplikacija dovedemo do pravljenja klijnet/server aplikacija.

Kada smanjimo količinu mrežnog saobraćaja tako što obradu upita prebacimo na back-end, Access postaje mnogo moćnije razvojeno rešenje. Tada Access može obraditi velike količine podataka i veliki broj konkurentnih korisnika. Osnovna pitanja sa kojima se programeri koji žele da naprave takve Accessove aplikacije obično suočavaju su:

- Raznolikost operativnih sistema koje korisnici mogu da koriste.
- Problemi u pravljenju aplikacije.
- Način na koji svaki korisnik uspostavlja vezu sa aplikacijom.
- Vrsta hardvera svakog korisnika.

Iako se obrada upita u klijent/server aplikaciji obavlja na serveru, čime se značajno smanjuje mrežni saobraćaj, sama aplikacija se ipak mora nalaziti u memoriji računara svakog korisnika. To znači da svaki klijentski računar mora da bude takav da se na njemu može izvršavati odgovarajući operarivni sistem i odgovarajuća verzija Accessa. Čak i kada se na klijentskom računaru izvršava odgovaraujući operativni sistem i kada na njemu postoji odgovarajuća verzija Accessa, i dalje mogu nastati problemi. DLL (Dynamic Link Libraries) konflikti su često rezultat grešaka koje se teško otkrivaju i idiosinkrazija koje se javaljaju u Accessovoj aplikaciji. Osim toga, Access nije najbolje rešenje za korisnike koji nemaju stalnu vezu sa bazom podataka, a koji preko Interneta moraju uspostaviti vezu sa aplikacijom i njenim podacima. Najzad, Access 11 ima velike zahteve za hardverom! To znači da, pre nego što odlučimo da prosledimo veliku Accessovu aplikaciju, moramo znati softversku i hardversku konfiguraciju svih sistemskih korisnika.



2.1.6 Access kao razvojna platforma za aplikacije u lokalnoj mreži/Internetu

Koristeći stranice za rad sa podacima, korisnici Interneta i lokalne mreže mogu ažurirati podatke aplikacije pomoću pretraživača (browser). Stranice za rad sa podacima su HTML dokumenti koji su direktno povezani sa podacima u bazi podataka. Iako Access stranice za rad sa podacima zapisuje van baze podataka, mi ih koristimo na isti način na koji koristimo standardne Accessove formulare, izuzev što ih je Microsoft dizjanirao tako da se izvršavaju u Microsoft Internet Exploreru 5.5 ili boljoj verziji, a ne u Microsoft Accessu. Stranice za rad sa podacima za obavljanje posla koriste dinamički HTML. Pošto se mogu koristiti samo u Microsoft Internet Exploreru 5.5 ili boljoj verziji, stranice za rad sa podacima su mnogo pogodnije rešenje za lokalne mreže nego za Internet.

Pored korišćenja stranica za rad sa podacima, objekte baze podataka možemo izdavati kao statičke ili dinamičke HTML stranice. Statičke stranice su standardni HTML dokumenti čiji se sadržaj može prikazati u bilo kom pretraživaču.

2.2 Access kao proizvod promenljive veličine

Jedna od jačih strana Accessa jeste promenljiva veličina njegovih aplikacija. Aplikaciju možemo prvo napraviti kao poslovnu aplikaciju koja se izvršava na jednom računaru pa je zatim proširiti u klijent/server aplikaciju firme. Ukoliko pravilno strukturiramo aplikaciju, postupak njenog proširivanja možemo obaviti tako da je malo ili nimalo ne prepravljamo. Ova osobina čini Access odličnim izborom za firme koje proširuju svoje poslovanje, kao i za aplikacije koje testiramo na nivou odeljenja imajući pri tom na umu da ih kasnije distribuiramo na nivou cele firme.

Ono što Access čini izuzetnim jeste da, čak i kada se koristi kao front end i back-end sa podacima u Accessovim tabelama na serveru datoteka, omogućava izuzetnu bezbednost i mogućnost uspostavljanja pravila baze podataka koja su se ranije mogla koristiti samo u back-end bazama podataka.



3 ŠTA JE SVE NOVO U ACCESSU 2003?

U Accessu 2003 postoje brojne nove osobine i svim tim osobina se vredi pozabaviti. Iako je Microsoft većinu novih osobina namenio klijent/server i web integraciji, postoje i mnoga druga korisna poboljšanja

3.1 Nove osobine formulara

Videćemo da u Accessu postoji nekoliko novih osobina formulara. Tri nova događaja olakšavaju rad sa formularima. Ti novi dogadaji su dogadaji OnUndo na nivou formulara i na nivou kontrola i događaj OnDirty. Dogadaj OnUndo na nivou formulara se dešava kada korisnik poništi sve izmene koje je načinio u formularu. Jednako kao parnjak na nivou formulara, tako se i događaj OnUndo na nivou kontrole dešava kada korisnik poništi izmene u kontroli. Najzad, kontrole sada imaju događaj OnDirty. To znači da možemo reagovati kada je kontrola "prljava", baš kao što možemo reagovati kada je formular "prljav".

Pored svih novih dogadaja formulara, u Accessu 2003 postoji novo svojstvo i nov metod. Novo svojstvo je svojstvo Movable. Ovo svojstvo postoji kako za formulare tako i za kontrole. Možemo ga korisriti kako bismo saznali da li se formular ili kontrola mogu pomerati. Novi metod je metod Move. Pomoću ovog metoda možemo programski pomerati formular.

Čarobnjaci ComboBox i ListBox u Accessu 2003 nam omogućavaju da zadamo redosled sortiranje po najviše četiri polja za podatke u poljima tipa ComboBox i ListBox. Na ovaj način ne moramo obavljati dodatni korak koji je bio neophodan u prethodnim verzijama Accessa.

U mnogim slučajevima osobina AutoCorrect nije podešna za podatke koji se nalaze u bazi podataka. U Accessu 2003 možemo bolje kontrolisati ponašanje osobine AutoCorrect. Na ovaj način ona postaje robusniji i korisniji alat u radu sa podacima tabele. Na primer, korišćenjem dugmeta AutoCorrect Options, veoma lako možemo poništiti ispravke ili uključiti i isključiti opciju AutoCorrect.

Sledeća nova osobina Accessa 2003 se odnosi na korišćenje tema Windowsa XP. Unapred je definisano da Access za formulare koristi izgled koji koristi operativni sistem. U Accessu 2003 postoji osobina koja nam omogućava da sprečimo da formulari naslede izgled od operativnog sistema.

Microsoft je u Access 2003 ugradio jednu grupa osobina, koja se opisuje kao pravila izgleda formulara i izveštaja, obezbeđuje proveru grešaka u formularima i izveštajima. Kada se aktiviraju, ove osobine ističu probleme sa nepridruženim oznakama (labelama) i kontrolama, novim nepridruženim labelama, greškama tastaturnih prečica (recimo, kada dve kontrole imaju istu tastaturnu prečicu) i nepravilnim svojsrvima kontrole.

Najzad, Microsoft je prilikom dizajniranja značajno unapredio rad sa podformularima u Accessu 2003. Najvažnije od svega je to što sada kontrolu podformulara možemo kliknuti desnim tasterom miša i odabrati View, Subform in Own Window. Ova nekada neophodna komanda otvara podformular u posebnom prozoru u pogledu Design.



3.2 Nove osobine izveštaja

Baš kao što za formulare postoji nekoliko novih osobina, tako i za izveštaje postoje nove osobine.

U Accessu 2003 postoji nekoliko novih svojstava. Ta svojstva obuhvataju svojsrva Modal, PopUp, BorderSryle, AutoResize, AutoCenter, MinMaxButtons, CloseButton i ControlBox. Sva ova svojsrva se ponašaju kao njihovi parnjaci za formulare i služe da izveštajima daju fleksibilnost kakvu imaju formulari.

Microsoft je metodu OpenReport dodao dva nova argumenta. Argument WindowMode omogućava korisniku da kontroliše ponašanje prozora izveštaja (na primer, da bude neprioritetan). Argument i svojstvo OpenArgs nam omogućavaju da izveštaju lako prosleđujemo informacije, kao da je otvoren, i da iz izveštaja možemo pročitati te informacije.

Kao što je Microsoft formularima dodao svojstvo Movable i metod Move, tako ih je učinio dostupnim i za izveštaje. Ove osobine nam omogućavaju da saznamo da li se izveštaj može pomerati i da na lak način napišemo kod za pomeranje prozora izvestaja.

3.3 Uzbudljivi svet pogleda PivotTable i PivotChart

Access 2003 omogućava korisniku da prikaže svaku tabelu, upit, formular, ADP tabelu, ADP pogled, ADP uskladištenu proceduru Hi ADP funkciju u pogledu PivotTable ili PivotChart.

Pogledi PivotTable i PivotChart omogućavaju korisniku da lako obavi prilično složene analize podataka. To znači da u Accessu možemo obaviti mnoga složena analiziranja podataka koja su se nekada mogla obaviti u Microsoft Excelu. Pogledi PivotTable i PivotChart se mogu koristiti u stranicama za rad sa podacima i podformularima, i možemo programski reagovati na dogadaje koje iniciraju.

3.4 Access 2003 i XML

Microsoft je u Access 2003 ugradio značajnu podršku za XML. Koristeći Microsoft Access, u XML možemo lako izvoziti podatke i lako ih možemo uvoziti iz XML-a, bilo programski, bilo pomoću korisničkog inlerfcjsa. Access podatke i šeme izvozi kao snimke stanja stalnih tabela ili upita.

Ono što je novo u Accessu 2003 jeste mogućnost zadavanja datoteke transformacije pri-likom uvoženja XML podataka ili izvoženja Accessovih podataka u XML. XML podatke možemo transformisati prilikom njihovog uvoženja. Transformisanje se obavlja pre nego što Access napravi novu tabelu ili uveze podatke u novu tabelu. Kada podatke izvozimo u XML, možemo opciono izvoziti XSL datoteku. Access generisanu XSL datoteku zasniva na informacijama o prezentovanju koje pronalazi u Accessovom formularu, izveštaju ili tabeli koju izvozimo.



3.5 Nove osobine ADP projekata

Mehanizam SQL Server 2000 Desktop Engine je zamena za mehanizam koji je ranije bilo poznat kao Microsoft Database Engine (MSDE). Ovaj robusni mehanizam ima iste osobine koje je Microsoft prvi put ugradio u SQL Server 2000.

3.6 Nove osobine stranica za rad sa podacima

Stranice za rad sa podacima u Accessu 2000 su imale ograničene osobine i sa njima je bilo teško raditi. Microsoft ih je značajno poboljšao u Accessu 2003. Na primer, dizajner Data Access Page Designer u Accessu 2003 podržava više niova poništavanja i ponovne primene. Ovaj dizajner nam takođe omogućava da odaberemo i radimo sa više objekata pomoću miša ili tastature. Microsoft je ugradio mnoge opcije za promenu veličine i prevlačenje u novi i poboljšani dizajner. U dizajeneru možemo koristi desni taster miša, a prozor Properties sada sadrži svojstva koja se mogu koristiri za odabrani objekat ili objekte. Kratko rečeno, dizajneri Accessa 2003 su veoma slični dizajnerima formulara i nisu više kao zapostavljena deca.



4 KAKO DA POČNEM DA PRAVIM ACCESSOVU APLIKACIJU?

Mnogi programeri veruju da, s obzirom na to da je Access okruženje za rapidno pravljenje aplikacija, ne postoji potreba za sistemskom analizom prilikom pravljenja aplikacije. Ne mogu opisati koliko se ne slažem sa takvim verovanjem. Kao što je rečeno ranije u ovom radu, Accessove aplikacije se varljivo lako prave, ali bez pravilnog planiranja mogu postati katastrofalne.

4.1 Analiza zadatka

Prvi korak u postupku pravljenja aplikacije jeste analiza zadataka, odnosno uzimanje u obzir svakog pojedinačnog procesa koji se odvija u svakodnevnom radu korisnika, dosadan je, ali neophodan posao.

4.2 Analiza i struktura podataka

Pošto smo analizirali i dokumentovali sve zadatke koji postoje u sistemu, spremni smo da se posvetite fazi analize i strukturiranja podataka aplikacije. U ovoj fazi moramo identifikovati svaki deo informacije koji je neophodan za obavljanje svakog zadatka. Te elemente podataka moramo pridružiti subjektima, a svaki subjekat će postati zasebna tabela u bazi podataka. Na primer, subjekat može biti klijent; svaki podatak koji se odnosi na tog klijenta - ime, adresa, telefonski broj, limit kredita i bilo koja druga značajna informacija -postaće polja u tabeli klijenata.

Za svaki podatak moramo da utvrdite sledeće:

- Odgovarajud tip podataka
- Neophodnu veličinu
- Pravila provere tačnosti podataka

Takođe, treba da utvrdimo da li ćemo korisniku omogućiti da ažurira svaki podatak i da li se podatak unosi ili izračunava. Zatim proveravamo da li smo pravilno normalizovali strukture tabela.



4.3 Testiranje

Bez obzira na trajanje, testiranja nikada nije dosta. Preporučujem da, ukoliko će se aplikacija izvršavati u operativnim sistemima Windows 98, Windows NT, Windows 2000 i Windows XP, testiranje obavimo u svim okruženjima. Takođe preporučujem da aplikaciju temeljno testiramo na najmanjem zajedničkom harverskom činiocu - aplikacija se može izvrsno izvršavati na našem računaru, ali može dati neprihvatljive rezultate na računarima korisnika.

4.4 Pravila imenovanja objekata

Uspostavljanje skupa pravila imenovanja - i njihovo poštovanje - jedan je od ključeva za uspešno programiranje u Accessu ili bilo kom drugom programskom jeziku. Kada biramo skup pravila imenovanja, obratimo pažnju na sledeće tri osobine:

- Lakoću korišćenja
- Čitljivost
- Koliko su rasprostranjena među programerima

Redikova pravila imenovanja, koje je predložio Greg Redik (Greg Reddick), svakako su najbolji skup pravila imenovanja koji postoji u svetu programiranja.

Redikova pravila imenovanja daju standardizovani način za imenovanje objekata. Ova pravila su izvedena iz Lesinski-Redikovih pravila imenovanja koja su predložena u Accessovim verzijama 1.x i 2.0. Ti standardi su usvojeni i korišćeni u svetu programiranja i mogu se pronaći u većini dobrih knjiga i časopisa napisanih u nekoliko proteklih godina.

Redik je revidirao pravila imenovanja kako bi ih prilagodio potrebama ljudi koji programiraju pomoću Accessa, Visual Basica, Excela i drugih Microsoftovih proizvoda koji koriste programski jezik VBA. Ova pravila su dosledna metodologija koju je lako koristiti za imenovanje objekata u svim tim okruženjima.



5 UPOZNAJMO SE SA ACCESS - om

Access je sistem za upravljanje bazama podataka (data base management system DBMS), koristi se za skladištenje i učitavanje podatke, prikazivanje, a takođe i za automatizaciju poslova koji se često ponavljaju. Predstavlja odličnu kombinacija produktivnosti programa za upravljanje bazom podataka i lakoće korišćenja Windows-a.

Kao zaokružena aplikacija, Access obezbeđuje potpuno upravljanje relacionom bazom podataka i integritet na nivou same mašine na kojoj se nalazi baza (što sprečava ne potpuno ažuriranje i brisanje). Tabelama je moguće pridružiti pravila koja proveravaju ispravnost i validnost podataka i sprečavaju unos neispravnih, a svako polje tabele može imati svoj format i podrazumevanu vrednost što znatno povećava produktivnost.

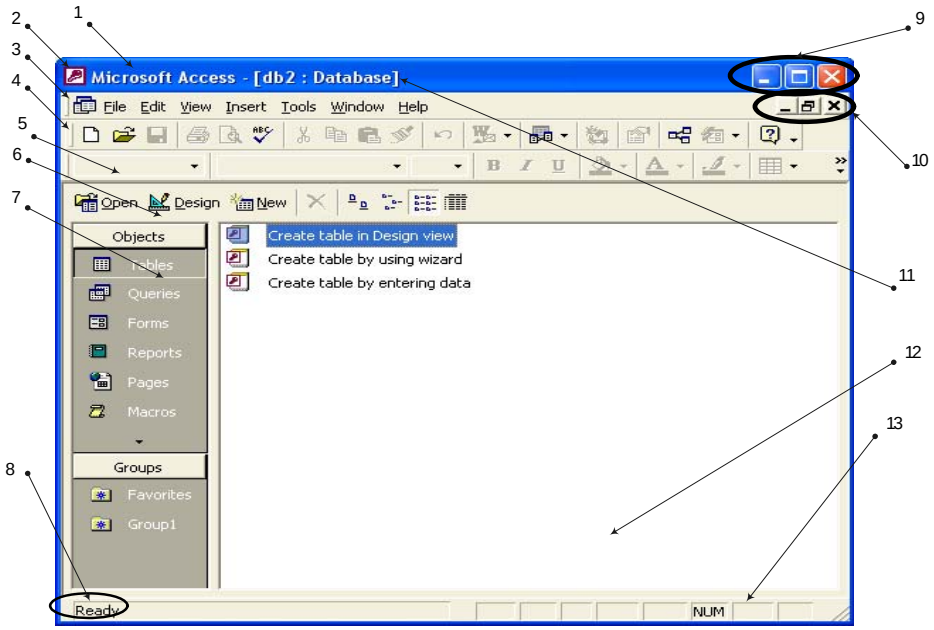
Omogućen je uvoz i izvoz podataka iz mnogih aplikacija: **dBase**, **FoxPro**, **Excel**, **SQL Server**, **Oracle**, **Btrieve**, tekstualni ASCII formati (polja sa fiksnom širinom i graničnicima), kao i podataka u HTML formatu. Access 2000 omogućava korišćenje svojih aplikacija za rad na Internetu. Izborom određene opcije omogućava se prebacivanje tabela, upita, izveštaja i prikaza u tabelarnom obliku u HTML format. Pomoću **Data Acess Pages** (stranice za pristup podacima) mogu da se prave složeni obrazci na Internetu koji su povezani sa podacima u konkretnoj mreži.

Za razliku od drugih programa za rad sa bazama podataka kod kojih se svaka tabela nalazi u posebnoj datoteci **(najčešće .dbf tipa)**, kompletna Access baza se nalazi u tačno jednoj datoteci (datoteka ima ekstenziju MDB, odnosne MDE ili ADP).

Tabele, upiti, formulari, izveštaji, stranice za rad sa podacima, makroi i moduli zajedno čine Accessovu bazu podataka. Svaki od ovih objekata ima specijalnu namenu. Accessova aplikacija takođe sadrži nekoliko različitih objekata, uključujući zavisnosti, komandne linije, svojstva baze podataka i uvozne/izvozne specifikacije. Pomoću ovih objekata možemo napraviti moćne, prijateljske, integrisane aplikacije.

6 OSNOVNI ELEMENTI ACCESS - a

Kada pokrenemo MS Access 2000 ili noviji na ekranu se pojavljuje radno okruženje koje možete videti na slici 1. Ekranom dominira prozor baze podataka u kojoj će mo izvoditi većinu operacija vezanih za našu bazu, a pored nje:



Slika 1: Izgled osnovnog prozora Access-a

1. **naslovna linija prozora** – sadrži ime dokumenta i ime programa;
2. **kontrolni meni dokumenta**;
3. **linija menija** – sadrži punu listu raznih Access-ovih komandi;
4. **paleta sa alatima Database**;
5. **paleta sa alatima Formatting** (**Datasheet**);
6. **meni baze** – sadrži opcije vezane za rad da bazom;
7. **meni Objects** – osnovni objekti;
8. **indikator statusa**;
9. **dugmad za promenu veličine i zatvaranje Access-a;**
10. **dugmad za promenu veličine i zatvaranje Access baze;**
11. **naziv baze;**
12. **prozor baze podataka;**
13. **statusna linija sa indikatorima;**



Access kao i svi **Microsoft**-ovi programi, raspolaže sa moćnim **Help** sistemom za pomoć koji može da bude koristan i početnicima i iskusnim programerima. Pomoć se dobija pritiskom na taster F1, a sadržaj zavisi od konteksta - elementa sa kojima se trenutno radi. Pored klasične pomoći na raspologanju je i **Screen Tips** koji prikazuju na ekranu kratke opise funkcije ili namene pojedinih objekata.

Korisnicima su na raspolaganju i **čarobnjaci (wizards)**, koji mogu drastično skratiti vreme neophodno za obavljanje nekog posla i svesti više satni rada na nekoliko minuta. Oni u nekoliko koraka postavljaju specifična pitanja vezana za sadržaj, stil ili format, a zatim automatski prave objekat. U Accessu postoji oko sto čarobnjaka za projektovanje baze podataka, aplikacija, tabela, obrazaca, izveštaja, dijagrama, poštanskih nalepnica, kontrolnih objekata i njihovih svojstava. Koriste se veoma često, a najviše prilikom projektovanja standardnih delova baze podataka.

Obrasci (Forms) i **izveštaji** se projektuju u **WYSIWYG** (što vidiš to dobiješ) okruženju i kada se dobije novi objekat odmah se vidi kako taj obrazac menja izgled. U obrascima i izveštajima može da se koristi veliki broj objekata koji omogućavaju postavljanje natpisa, unošenje podataka, kartice, opcionu dugmad, polja za potvrđivanje, izbor podataka iz listi, crtanje linija, pravougaonika, menjanje boja i senčenja, korišćenje slika, dijagrama, podobrazaca i podizveštaja. Obrazci mogu imati više strana, a izveštaji više nivoa grupisanja i međuzbirova. Veoma praktična alatka pri izradi izveštaja je Report Writer i ona omogućava jedan od deset različitih nivoa međuzbirova i sortiranja.

Uspostavljanje **relacija (veza)** između pojedinih tabela u Acessu moguće je grafičkim putem u okviru prozora **Relationship**. Tu se vrši povezivanjei tabele različitih formata, koje kad se jednom povežu ponašaju kao jedinstvena celina. Takođe mogu se izdvajati pojedinačna polja, definisati redosled sortiranja, sastavljati izračunati izrazi i zadavati više ili manje složeni uslovi za izdvajanje pojedinih zapisa. Rezultati pojedinih upita mogu se prikazivati putem obrazaca (na ekranu) ili izveštaja (na štampaču).

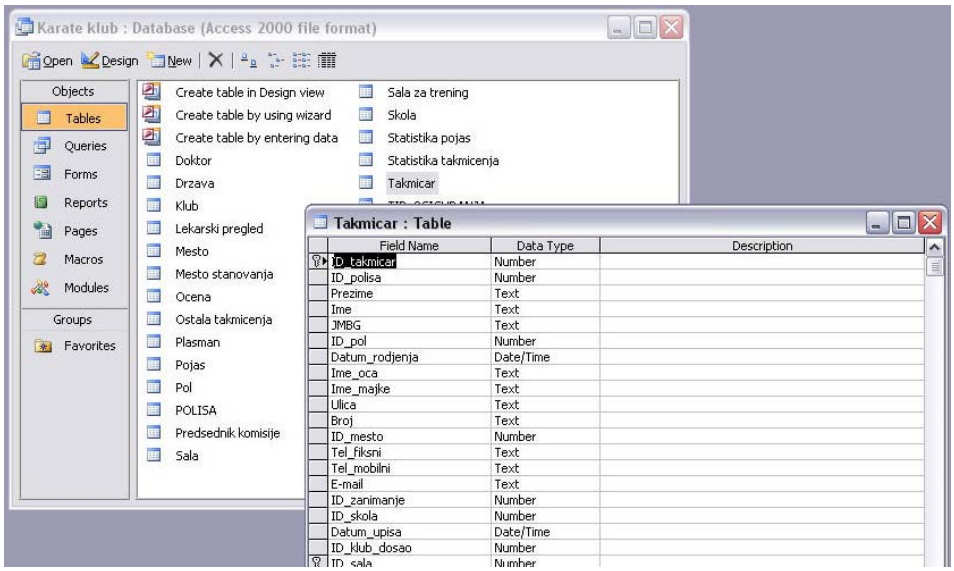
Sve alatke za izradu grafikona koje su ugrađene u **MS Word**, **Excel** ili **Power Point** mogu da se pronađu i u Access-u. Može da se koristi neograničen broj različitih tipova poslovnih grafikona gde se njihov izgled prilagođava svim potrebama, mogu biti povezani sa obrascima i tako se dobija moćan grafički prikaz podataka koji se menja u zavisnosti od trenutnog zapisa.

U Acess su ugrađene **funkcije** koje rade sa objektima baze podataka, matematičke funkcije, poslovne funkcije, finansijske funkcije, funkcije za rad sa datumima i vremenima, kao i funkcije za rad sa tekstualnim nizovima. One mogu da se koriste u obrazcima, izveštajima i upitima da bi se sastavili izračunati izrazi. Za one koji ne žele da programiraju obezbeđeni su **makroi**. Oni omogućavaju obavljanje uobičajenih poslova bez intervencije korisnika, postojie skoro 50 makro akcija koje omogućavaju rad sa podacima, pravljenje menija i okvire za dijalog, otvaranje obrazaca i izveštaja i automatizaciju bilo kog posla.

7 OBJEKTI BAZE

Razvijena baza podataka u Accessu se sastoji od sledećih elemenata, tj. objekata: tabele, upiti, maske i izveštaji; a može sadržati: Data Access strane, makroe i/ili VBA module (Visual Basic for Application - programski jezik za programiranje operacija u Office aplikacijama).

8 TABELE: Skladište podataka



Slika 2: Izgled prozora za kreiranje tabela

Tabele su nosioci podataka. Podaci se mogu nalaziti isključivo u tabelama. Da bismo saznali koje se sve tabele nalaze u otvorenoj bazi podataka, kliknimo ikonu Tables u listi Objects. Ukoliko želimo da vidimo podatke koji se nalaze u tabeli, dva puta kliknimo ime tabele čije podatke želimo da vidimo.

Access podatke tabele prikazuje u tabelarnom obliku, što znači da se prikazuju sva polja i slogovi tabele. Možemo menjati mnoge atribute tabelarnog prikaza, a možemo čak i da vršimo pretraživanje i filtriranje podataka u tabeli. Ukoliko je sa tabelom povezana druga tabela, možemo otvoriti i zatvoriti podtabelu kako bismo videli podatke koji se nalaze u dete-tabelama.

Najčešće će nas kao programere zanimati struktura tabele koja je zapravo njen šematski plan ili šablon. Da bismo videli strukturu tabele, kliknimo ikonu Design kada je tabela označena. U pogledu Design možemo videti ili izmeniti sva imena polja, tipove podataka i svojstva polja i tabele. Access nam daje mod i fleksibilnost koji su nam neophodni za prilagođavanje strukture tabela

Svaka tabela se sastoji od kolona i redova, dok se presek kolone i reda naziva **polje.** Polje može biti sledećeg tipa: tekst, broj, datum/vreme, logičko polje, memo polje (polje u



koga se može uneti tekst proizvoljne veličine), polje za unos brojeva u novčanom formatu, posebno polje u kome se automatski po započetom unosu sloga inkrementira broj (polje **AutoNumber**), hiperlink polje (pokazivač na adresu na Internetu), OLE objekt polje (polje sa objektom povezanim iz drugog programa) i **Look Up** polje (polje čiji prikazani sadržaj se nalazi u drugoj tabeli). Pri kreiranju baze podataka veoma je bitno postaviti tip polja, jer su brzina rada, organizovanost podataka i veličina same baze u direknoj vezi sa ispravno postavljenim tipom polja. Npr. za datum tipa **01/01/2005** tip polja ćemo postaviti na **Date/Time** (datum/vreme); za isključivo numeričke vrednosti postavljamo tip polja na **Number**; za kombinovani alfanumerički unos tip polja postavljamo na **Text** kod koga se predviđa maksimalni broj karaktera u unosu, itd.

Baza podataka može (često i treba) da sadrži više tabela koje se preko relacija međusobno povezuju. Npr. članove kluba ćemo držati u jednoj tabeli, pojaseve u drugoj, podake o takmičenjima u trećoj itd. Relacije se uspostavljaju samo među istovetnim podacima. Na ovaj način se izbegava višestruki unos podataka.

8.1 Odabiranje odgovarajućeg tipa polja za podatke

Tip podataka koji biramo za svako polje može značajno uticati na performanse i funkcionalnost aplikacije. Nekoliko činilaca utiče na izbor tipa podataka za svako polje tabele:

- Tip podataka koji se zapisuje u polju
- Da li će se sadržaj polja koristiti za izračunavanja
- Da li je potrebno da podatke polja sortiramo
- Način na koji želimo da sortiramo podatke polja
- Koliko nam je bitan prostor za smeštanje podataka

Tip podataka koji će se zapisivati u polja najviše utiče na tip podataka koji ćemo odabrati. Na primer, ukoliko u polje treba da zapišemo brojeve koji imaju vodeće nule, ne možemo odabrati tip Number jer će Access u polju ignorisati vodeće nule. Ovo pravilo se odnosi na podatke kakvi su poštanski broj (neki počinju nulama) i kodovi odeljenja.

Ukoliko se sadržaj polja koristi za izračunavanje vrednosti, moramo odabrati tip podataka Number ili Currency. Izračunavanja ne možemo obavljati sa podacima koji se nalaze u poljima za koja je definisan neki drugi tip podataka. Jedini izuzetak od ovog pravila jeste polje za koje je definisan tip podataka Date. Polja tipa Date se mogu koristiti za izračunavanje datuma/vremena.

Takođe moramo voditi računa da li ćemo podatke koji se nalaze u polju koristiti za pravljenje indeksa ili za sortiranje. Polja tipa Hyperlink ili OLE ne možemo koristiti za sortiranje, pa ove tipove podataka nemojmo odabrati ukoliko podatke iz polja moramo koristiti za pravljenje indeksa ili sortiranje. Osim toga, moramo razmisliti o načinu na koji ćemo sortirati podatke. Na primer, u polju tipa Text, Access će skup brojeva sortirati prema redosledu znaka koji se nalazi na mestu najveće težine, zatim po sledećem mestu, i tako dalje (to jest, 1,10, 200, 2, 20, 200) jer se za polja tipa Text u Accessu koristi standardna ASCII sekvenca. Sa druge strane, Access će brojeve u polju tipa Number ili Currency sortirati u rastućem redosledu (to jest, 1,2,10,20,100,200). Možda mislimo da nikada neće biti potrebno da podatke sortiramo po standardnoj ASCII sekvenci, ali za neke



vrste informacija ponekad ima smisla tako sortirati. U Accessu 2003 postoji mogućnost sortiranja ili grupisanja po polju tipa Memo, ali se sortiranje i grupisanje obavlja samo po prvih 255 znakova. Najzad, moramo razmisliti koliko nam znači prazan prostor na disku. Svaki tip podataka zauzima različit prostor na hard disku, što može biti faktor pri biranju tipa podataka za polje.

U Accessu za polje možemo upotrebiti jedan od devet tipova podataka. To su: Text, Memo, Number, Date/Time, Currency, Yes/No, OLE Object i Hyperlink. U tabeli 1 se nalaze informacije o prostoru na hard disku koji zauzima svaki od ovih tipova i za kakve podatke je pojedini tip podataka pogodno koristiti.

Tabela 1: Prostor na disku koji zauzimaju Accessovi tipovi podataka i kada je pogodno koristiti ih.

Tip polja (podataka)	Pogodan za	Prostor koji zauzima
Text	Podaci koji sadrže tekst, kombinacija teksta i brojeva ili brojevi koje nećemo koristiti za izračunavanja. Primeri su: imena, adrese, kodovi odeljenja i telefonski brojevi.	Zavisi od onoga što se zapisuje u polje, pa može zauzeti od 0 do 255 bajtova.
Memo	Veliki tekst i numerički stringovi. Primeri su beleške i opisi.	Može zauzeti od 0 do 65.536 bajtova.
Number	Podaci koji se koriste za izračunavanja (ne uključujući novac). Primeri su: godine starosti, kodovi, recimo identifikator klijenta ili načini plaćanja.	Može zauzeti 1, 2, 4 ili 8 bajtova što zavisi od odabrane veličine polja (ili 16 bajtova za replikaciju identifikatora).
Date/Time	Datumi i vremena. Primeri su datum naručivanja i datum rođenja.	Zauzima 8 bajtova.
Currency	Vrednosti valuta. Primeri su dug i cena.	Zauzima 8 bajtova.
Auto-Number	Jedinstveni uzastopni ili slučajno odabrani brojevi. Primeri su brojevi faktura i brojevi projekata.	Zauzima 4 bajta (16 bajtova za replikaciju identifikatora).
Yes/No	Pogodan je za polja koja mogu imati jednu od dve moguće vrednosti (da/ne, tačno/netačno). Primeri korišćenja su označavanje da li je račun plaćen ili stanje imovine.	Zauzima 1 bit.
OLE Object	Koristi se za smeštanje objekata kao što su dokumenti Worda ili Excela.	Zauzima od 0 bajtova do 1 gigabajta, što zavisi od veličine podataka koji su zapisani u polju. Primeri su podaci o zaposlenima ili budžetu.
Hyperlink	Tekst, ili kombinacija teksta ili brojeva, koji se zapisuje kao tekst i koristi kao hiperlink za web adrese (URL) ili UNC putanju. Primeri su web stranice ili datoteke u mreži.	Zauzima od 0 do 2048 bajtova za svaki od tri dela koji sačinjavaju adresu (ukupno do 64.000 znakova).
Lookup Wizard	Koristi se za pravljenje polja koje korisniku omogućava da iz druge tabele odabere vrednost ili da vrednost odabere iz spiska polja tipa Combo za koje nam čarobnjak pruža pomoć.	Obično zauzima 4 bajta. Zauzima prostora koliko i primarni ključ za polje tipa Lookup.

Najteži deo odabiranja tipa polja jeste u saznavanju koji tip polja je najbolje upotrebiti u svakoj situaciji. Detaljno razmatranje svakog tipa polja koje sledi i kada treba da ga upotrebimo će nam pomoću prilikom odlučivanja.

8.1.1 Polja tipa Text: Najčešći tip polja

Najveći broj polja je tipa Text. Mnogi programeri ne shvataju da je polja tipa Text najbolje koristiti za sve brojeve koji se ne koriste za izračunavanja. Primeri su telefonski brojevi, šifre delova i poštanski brojevi. Iako je unapred određena veličina polja tipa Text 50 znakova, u polje ovog tipa možemo smestiti najviše 255 znakova. Pošto Access dinamički rezerviše prostor na disku, velika polja ne zauzimaju prostor na disku, ali performanse možemo poboljšati ukoliko rezervišemo najmanju moguću veličinu polja. Svojstvo FieldSize koristimo za zadavanje maksimalnog broja znakova koji se može smestiti u polje tipa Text.



8.1.2 Polja tipa Memo: Za velike beleške i komentare

U polja tipa Memo možemo smestiti najviše 65.536 znakova, što znači da može sadržati 16 stranica teksta u svakom slogu. Polja tipa Memo su odlična za svaku vrstu beleški koje želimo da zapišemo uz podatke tabele. Zapamtimo, u Accessu 2003 možemo vršiti sortiranje po polju tipa Memo.

8.1.3 Polja tipa Number: Za slučajeve kada treba obaviti izračunavanja

Polja tipa Number koristimo za zapisivanje podataka koje moramo koristiti za računanje drugih vrednosti. Ukoliko prilikom izračunavanja moramo da koristimo novčane vrednosti, umesto polja tipa Number treba da koristimo polje tipa Currency. Polje tipa Number je zapravo nekoliko tipova polja u jednom jer u Accessu 2003 postoji sedam različitih veličina numeričkog polja. U polje tipa Number veličine Byte možemo smestiti pozitivne cele brojeve od 0 do 255, veličine Integer možemo smestiti cele brojeve od -32768 do 32767, a veličine Long Integer možemo smestiti cele brojeve od manje od -2 milijarde do više od 2 milijarde. Iako sve tri veličine polja tipa Number imaju odlične performanse, svaki tip zahteva značajno veći prostor za zapisivanje. Druge dve veličine numeričkog polja, veličine Single i Double, služe za rad sa brojevima u pokretnom zarezu i, stoga, mnogo lošije performanse. U polje tipa Number veličine Single možemo zapisati razlomljene brojeve sa najviše sedam značajnih cifara. U polje tipa Number veličine Double možemo zapisati razlomljene brojeve sa najviše četrnaest značajnih cifara. Tip Decimal je tip podataka koji se prvi put pojavio u Accessu 2002. Tip podataka Decimal omogućava zapisivanje veoma velikih brojeva i obezbeđuje preciznost do 28 cifara! Poslednja veličina polja tipa Number, veličina Replication ID, obezbeđuje jedinstveni identifikator pomoću postupka sinhronizacije podataka.

8.1.4 Polja tipa Date/Time: Za evidentiranje događaja

Polje tipa Date/Time koristimo za zapisivanje datuma i vremena. Polja tipa Date/Time nam omogućavaju da obavimo izračunavanja sa datumima i da obezbedimo da su datumi i vremena uvek pravilno sortirani. Access zapravo datum i vreme interno zapisuje kao 8-bajtni broj u pokretnom zarezu. Vreme se u Accessu zapisuje kao razlomljeni deo dana.

Access na podatke preslikava podešavanja datuma i vremena koje uspostavimo u Windowsovom Control Panelu. Na primer, ukoliko u Control Panleu u Regional Settingsima promenimo oblik za Short Date, u formularima, izveštajim i tabelama će se odmah odslikati načinjene promene.

8.1.5 Polja tipa Fields: Zapisivanje novčanih vrednosti

Polje tipa Currency je specijalan tip numeričkog polja koji koristimo kada u tabelu zapisujemo novčane vrednosti. Polje tipa Currency ne dozvoljava zaokruživanje podataka prilikom izračunavanja. U ovim poljima se zapisuje 15 cifara celog dolara i pride tačnost na stoti deo centa. Iako je veoma precizno, ovaj tip polja se veoma sporo obrađuje.



Access na podatke preslikava sva podešavanja formata novčanih vrednosti koje uspostavimo u Windowsovom Control Panleu. Naravno, u Accessu se automatski ne obavlja konvertovanje iz jedne u drugu valutu. Ukoliko iz Control Panela u Regional Settingsima promenimo simbol valute, baš kao što važi za datume, te promene će se odmah odslikati u formularima, izveštajima i tabelama.

8.1.6 Polja tipa AutoNumber: Za jedinstvene identifikatore slogova

Kada korisnik unosi novi slog, Access automatski generiše vrednosti za polje tipa AutoNumber. U ranijim verzijama Accessa, vrednosti brojača su morale biti uzastopne. Polje tipa AutoNumber u Accessu 2003 može imati uzastopne vrednosti ili vrednosti koje se dobijaju slučajnim izborom. Slučajan izbor vrednosti treba koristiti kada nekoliko korisnika, koji nemaju uspostavljenu vezu sa bazom podataka, unose slogove jer je malo verovatno da će Access generisati istu vrednost za dva razlidta sloga. Specijalni tip polja AutoNumber jeste tip Replication ID. Ovi jedinstveni brojevi dobijeni na slučajan način pomažu u postupku replikovanja tako što generišu jedinstvene identifikatore koji se koriste za sinhronizovanje replika baze podataka.

O poljima tipa AutoNumber u kojima se nalaze uzastopni brojevi treba da zapamtimo nekoliko važnih osobina. Ukoliko korisnik iz tabele ukloni slog, jedinstveni broj tog sloga je zauvek izgubljen. Isto tako, ukoliko korisnik dodaje slog pa zatim poništi akciju, jedinstvena vrednost brojača tog sloga je zauvek izgubljena. Ukoliko ovakvo ponašanje nije prihvatljivo, sami možemo generisati vrednosti brojača.

8.1.7 Polja tipa Yes/No: Kada je jedan od dva odgovora tačan

Polja tipa Yes/No treba da koristimo za zapisivanje logičkih vrednosti tačno i netačno. Ono što se zaista zapisuje u polje jeste vrednost -1 za tačno (Yes), 0 za netačno (No) ili Null kada ništa nije odabrano. Format prikazivanja polja određuje šta će korisnik zapravo videti (obično Yes/No, True/False, On/Off ili treću mogućnost - Null - ukoliko svojstvu TripleState pridružene kontrole zadamo vrednost True). Polja tipa Yes/No su efikasna za sve podatke koji mogu imati samo vrednosti tačno ili netačno. Ne samo da korisniku dozvoljavaju da odabere samo jednu vrednost, već zauzimaju samo jedan bit.

8.1.8 Polja tipa OLE Object: Polje u koje možemo smestiti gotovo bilo šta

Microsoft je polje tipa OLE Object napravio za zapisivanje podataka iz bilo koje serverske aplikacije registrovane u Windowsu koja može da poveže i ugnjezdi objekte (OLE), uključujući radne tabele, dokumente programa za obradu teksta, zvuk i video. Postoji mnogo nadna za poslovno korišćenje polja tipa OLE Object, recimo za zapisivanje kratkih biografija, infromacija o zaposlenima, budžetu ili video zapisima. Međutim, u mnogim slučajevima, mnogo je efikasnije koristiti polje tipa Hyperlink za zapisivanje linka ka dokumentu umesto zapisivanja celog dokumenta u polje tipa OLE Object.



8.1.9 Polja tipa Hyperlink: Vaša veza sa Internetom

Polja tipa Hyperlink koristimo za zapisivanje adresa uniformnog lokatora resursa (URL) (to su linkovi ka web stranicama na Internetu) ili lokalnoj mreži, ili putanja univerzalnih pravila imenovanja (UNC) (linkovi ka lokacijama datoteka). Access polje tipa Hyperlink deli na tri dela:

- Ono što korisnik vidi
- URL ili UNC
- Podadresu, recimo ime opsega ili obeleživač

Pošto korisnik u polje tipa Hyperlink unese podatak, on služi kao direktna veza sa datotekom ili stranicom na koju se odnosi.

Pošto smo u tabeli napravili polja, potrebno je da podesimo njihova svojstva. Svojstva polja nam omogućavaju da kontrolišemo kako će Access zapisivati podatke kao i kakve podatke korisnik može da unese u polje. U zavisnosti od tipa polja zavisi koja svojstva ćemo moći da podesimo. Najveći spisak svojstava postoji za polje tipa Text. U narednim odeljcima se opsuju svojstva polja.

8.1.10 Svojstvo Field Size: Kako ograničiti unošenje u polje

Prvo svojstvo jeste svojstvo Field Size i postoji samo za polja tipa Text i Number. Kao što je rečeno, najbolje je svojstvu Field Size zadati najmanju moguću vrednost. Za polja tipa Number to znači najmanji prostor za zapisivanje i najbolje performanse.

8.1.11 Svojstvo Format: Način prikazivanja podataka

Drugo svojstvo jeste svojstvo Format i postoji za sve tipove polja osim za polja tipa OLE Object. Ovo svojstvo nam omogućava da zadamo kako će Access prikazivati podatke. Access nam omogućava da odaberemo jedan od predefinisanih svojstava ponuđenih formata ili da napravimo nov format prikazivanja podataka. Na primer, u Accessu možemo odabrati jedan od brojnih formata za datum/vreme, uključujući formate Short Date (7/7/07); Long Date (Sunday, July 7,2007); Short Time (7:17); i Long Time (7:17:11AM). Formati za polje tipa Currency obuhvataju formate Currency ($1,767.25); Fixed (1767.25); and Standard (1,767.25).

Zadajmo format Medium Date svojstvu Format polja ContactDate. Pređimo u pogled Datasheet i unesimo datume u različitim oblicima, recimo 07/04/07 and July 4, 2007. Videćemo da će se, bez obzira kako unesemo datume, čim predemo u drugo polje, prikazati u formatu dd/mm/yy, to jest 04-July-07.

8.1.12 Svojstvo Input Mask: Kakvi podaci se smeštaju u polje

Sledeće važno svojstvo jeste svojstvo Input Mask, a postoji za polja tipa Text, Number, Date/Time i Currency. Svojstvo Format utiče na prikazivanje podataka u Accessu, dok svojstvo Input Mask kontroliše kakve podatke Access zapisuje u polje.



Svojstvo Input Mask upotrebljavamo kako bismo kontrolisali, znak po znak, kakve znakove (numeričke, alfanumeričke, i tako dalje) Access može da zapiše i da li se zahteva neki određeni znak. Čarobnjak Input Mask Wizard nam pomaže da napravimo najčešće korišćene maske samo za polja tipa Text i Date. Da bismo pokrenuli čarobnjaka Input Mask Wizard, kliknimo dugme koje se nalazi desno od polja Input Mask.

Napomena

Čarobnjak Input Mask Wizard se može koristiti samo ako smo prilikom instaliranja odabrali komponentu Additonal Wizards. Ukoliko nismo, Access će nam predložiti da je instaliramo prvi put kada pokušamo da je upotrebimo.

Evo detaljnijeg primera: u masci \(999") "000\-0000;;_, prva obrnuta kosa crta prouzrokuje da se prvi znak koji je sledi (otvorena zagrada) prikazuje kao literal. Tri devetke omogućavaju korisniku da unese opcione brojeve ili razmake. Access prikazuje zatvorenu zagradu i razmak koji se nalaze između znaka navoda kao literale. Prve tri nule znače da se moraju uneti cifre od 0 do 9. Access prikazuje crticu iza sledeće obrnute kose crte kao literal. Zatim se od korisnika zahteva da unese još četiri cifre. Između dva znaka tačka - zarez nema ništa pa Access prikazuje podvlaku kako bi korisniku naznačio gde unosi sleded znak.

Upotrebimo čarobnjaka Input Mask Wizard za pravljenje maske za polje PhoneNumber koje bi trebalo da smo definisali kao polje tipa Text.

- Da bismo to uradili, kliknimo bilo gde u polju PhoneNumber, pa zatim kliknimo svojstvo Input Mask.
- Kliknimo tri tačke koje se nalaze desno od svojstva Input Mask.
- Iz spiska postojećih maski odaberimo Phone Number i kada nas čarobnjak upita "Kako želite da zapišete podatke?" (How do you want to store the data?), odaberimo da se literali ne zapisuju u polje.
- Pređimo u pogled Datasheet i unesimo telefonski broj. Videćemo kako kursor preskače literale. Mesto za pozivni broj ostavimo prazno. Access bi trebalo da nam to dozvoli.
- Sada na bilo kojoj poziciji pokušajmo da unesemo slovo - Access nam to neće dopustiti.
- Zatim, pokušajmo da ne unesemo bilo koju cifru sedmocifrenog telefonskog broja. Access nam ni ovo neće dopustiti.

Savet

Kada koristite masku unosa, korisnik je uvek u režimu Overtype. Ovakvo ponašanje je osobina Accessa i ne može se promeniti.

8.1.13 Svojstvo Caption: Velika ušteda vremena

Sledeće svojstvo koje možemo koristiti jeste svojstvo Caption. Tekst koji se nalazi u ovom svojstvu postaje natpis labele koja je pridružena kontroli za podatke kada ih pravimo na formularima ili izveštajima. Svojstvo Caption postaje značajno svaki put kada polja imenujemo tako da u imenu nema razmaka. Šta god se nalazi u svojstvu Caption zamenjuje ime polja koje se koristi u pogledu Design, za formulare i izveštaje.



Veoma je važno u svojstvo Caption polja zapisati informaciju pre nego što napravimo formulare ili izveštaje u kojima se polje koristi. Kada pravimo formular ili izveštaj, Access proverava tekući sadržaj svojstva. Ukoliko kasnije promenimo ili unesemo tekst u svojstvo, Access za to polje neće izmeniti natpise u formularima i izveštajima koji su već napravljeni.

8.1.14 Svojstvo Default Value: Uštedite vreme prilikom unošenja podataka

Sledeće važno svojstvo jeste svojstvo Default Value koje se koristi za definisanje osnovne vrednosti koju Access zapisuje u polje kada korisnik unosi nove slogove u tabelu. Ove unapred definisane vrednosti, koje mogu biti tekst ili izrazi, mogu osobi koja unosi podatke značajno uštedeti vreme. Međutim, Access ih ne koristi za proveru podataka koje korisnik unosi u polje.

8.1.15 Svojstvo Validation Rule: Kontroliše šta korisnik unosi u polje

Svojstvo Validation Rule predlaže korisniku kakvu vrednost treba da unese, međutim ovo svojstvo zapravo ograničava šta korisnik može da unese. Korisnik ne može da prekrši pravila. Mehanizam baze podataka ova pravila striktno primenjuje. Kao što je slučaj sa svojstvom Deafult Value, svojstvo Validation Rule može sadržati tekst ili ispravan Accessov izraz, ali u svojstvu Validation Rule ne može koristiti funkcije koje sami definišemo. Takođe, u svojstvu Validation Rule ne možemo koristiti reference na formulare, upite ili tabele.

Napomena

U ovom primeru se izraz <=Date() koristi kako bi se ograničila vrednost koja se unosi u polje na datum koji odgovara tekućem ili ranijem datumu. Pošto izraz Date() kao rezultat uvek daje tekući datum, pravilo važi kada korisnik unosi nov slog i kada menja postojeći.

8.1.16 Svojstvo Validation Text: Kako obezbediti poruku za korisnika

Svojstvo Validation Text koristimo za zadavanje poruke o grešci koju će korisnici videti kada se prekrše pravila provere. U svojstvu Validation Text se mora nalaziti tekst. U ovom svojstvu ne možemo koristiti izraze.

U svako od polja pokušajmo da unesemo vrednosti koje ne odgovaraju pravilima i obratimo pažnju na poruke koje će se prikazati.

8.1.17 Svojstvo Required: Naterajte korisnika da unese vrednost

Svojstvo Required je veoma važno - ovo svojstvo oderđuje da li se u polje mora uneti vrednost. Ovo svojstvo je korisno za polja stranog ključa (foreign key) kada želimo da



budemo sigurni da će se u polje uneti vrednost. Svojstvo nije korisno za polja koja sadrže poslovne informacije (na primer, ime kompanije).

8.1.18 Svojstvo Allow Zero Length: Prilagodite polje za nepostojeće podatke

Svojstvo Allow Zero Length je slično svojstvu Required. Ovo svojstvo korisrimo kako bismo odredili da li je korisniku omogućeno da unese prazan string (""). Prazan string nije isto što i vrednost Null (koja označava da ništa nije uneto). Ovakvi stringovi označavaju da za polje ne postoje podaci. Na primer, zaposleni, koji je stranac, možda nema broj socijalnog osiguranja. Unošenjem praznog stringa, osoba koja unosi podatke naznačava da broj socijalnog osiguranja ne postoji.

U ranijim verzijama Accessa je svojstvu Allow Zero Length unapred bila zadata vrednost No. Microsoft je u Accessu 2002 i Accessu 2003 promenio unapred zadatu vrednost u Yes. Budimo oprezni zbog ove promene, naročito ako smo navikli da radimo sa prethodnim verzijama Accessa.

Savet

Nemojmo zaboraviti da ukoliko želimo da poništimo izmene treba da jednom pritisnemo taster Esc. Da bismo poništili sve izmene sloga, dva puta pritisnimo taster Esc.

Svojstva Required i Allow Zero Length utiču jedno na drugo. Ukoliko je svojstvu Required zadata vrednost Yes, a svojstvu Allow Zero Length vrednost No, onda smo prema korisnicima postavili najstroža pravila za unošenje podataka. Ne samo da korisnici moraju uneti vrednost, već ta vrednost ne može biti prazan string.

Ukoliko smo svojstvima Required i Allow Zero Length zadali vrednost Yes, od korisnika zahtevajmo da unese vrednost, ali ta vrednost može biti prazan string. Međutim, ukoliko smo svojstvima Required i Allow Zero Length zadali vrednost Yes, korisnicima smo omogućili da unesu vrednost Null, ali im nismo omogućili da unesu prazan string.

Najzad, ukoliko je svojstvu Required zadata vrednost No, a svojstvu Allow Zero Length vrednost Yes, prema korisnicima smo postavili najblaža pravila za unošenje podataka. U ovom slučaju korisnici mogu uneti vrednost Null ili prazana string.

8.1.19 Svojstvo Indexed: Ubrzajte pronalaženje podataka

Indekse koristimo za poboljšavanje performansi kada korisnik želi da pronađe polje. Iako je u opštem slučaju bolje napraviti više nego manje indeksa, indeksi imaju mane. Opšte je pravilo da se indeksi prave za sva polja koja se redovno koriste za pronalaženje i sortiranje podataka, i kao kriterijum u upitima.

Pomoću indeksa se ubrazava pronalaženje, sortiranje i grupisanje podataka. Mana im je što zauzimaju prostor na disku i što usporavaju izmenu, dodavanje i uklanjanje podataka. Iako u većini slučajeva korist od indeksiranja nadmašuje nedostatke, ne treba da indeksiramo svako polje svih tabela. Indekse pravimo samo za polja, ili kombinaciju polja, po kojima će korisnik



obavljati sortiranje i pretraživanje. Nemojmo praviti indekse za polja koja sadrže podatke koji se ponavljaju, recimo za polje koje može sadržati samo dve različite vrednosti. Najzad, nikada nemojmo praviti indeks za polja tipa Yes/No. Ova polja zauzimaju samo jedan bit, pa stoga za njih važi prethodno pravilo jer mogu imati samo jednu od dve vrednosti. Zbog toga indeksi nisu korisni za polja tipa Yes/No.

Indeksi su jednako važni na serveru baze podataka. Kada Accessovu bazu podataka prebacujemo na server koji nije Microsoftov, prilikom prebacivanja se ne prave indeksi. U tom slučaju na back-end serveru baze podataka moramo ponovo napraviti indekse. Ukoliko se na serveru baze podataka izvršava Microsoft SQL Server, za prebacivanje Accessove baze podataka možemo u Accessu 2003 upotrebiti čarobnjaka Upsizing Wizard. Ovaj alat pravi indekse za serverske tabele ako indeksi postoje u Accessovim tabelama.

8.1.20 Svojstvo Unicode Compression: Komprimovanje podataka

Sledeće važno svojstvo jeste svojstvo Unicode Compression. Svojstvo Unicode Compression postoji samo za polja tipa Text i Memo. Ovo svojstvo koristimo kako bismo odredili da li podaci polja treba da se komprimuju pomoću Unicode sažimanja. U verzijama Accessa pre Accessa 2000 podaci su se zapisivali u DBCS (dvobajtni skup znakova (double-byte character set)) formatu koji je napravljen radi zapisivanja znakova nekih jezika, recimo kineskog. U Accessu 2000, Accessu 2002 i Accessu 2003 svi znaci se zapisuju u Unicodeovom dvobajtnom forma tu. Iako je u ovom formatu za zapisivanje svakog znaka potrebno više prostora (dva bajta umesto jednog bajta), svojstvo Unicode Compression omogućava komprimovanje podataka. Ukoliko skup znakova koji se koristi omogućava komprimovanje, a svojstvu Unicode Compression je zadata vrednost Yes, podaci kolone se zapisuju u komprimovanom formatu.

8.1.21 Korišćenje osobine Lookup

Koristeći čarobnjaka Lookup Wizard možemo zadati da polje svoje vrednosti pronalazi u drugoj tabeli ili upitu, ili iz spiska stalnih vrednosti. Spisak važećih vrednosti možemo prikazati u polju tipa ListBox ili ComboBox. Ovakvo polje se obično pravi od stranog ključa (foreign key) (strana "više") do primarnog ključa (strana "jedan") zavisnosti jedan-prema-više.

Napomena

Ukoliko odaberemo više polja za pronalaženje podataka, a jedno od njih je polje ključa, recimo polje ID, prikazaće se polje za potvrdu Hide Key Column. Ovo polje treba da ostane potvrđeno. Na taj način se automatski sakriva polje ključa iako će krajnji rezultat biti vezan to polje.

Napomena

Osobina pronalaženja podataka je više smetnja nego pomoć. Pošto pozovemo ovu osobinu, mi i naši korisnici više nećemo moći da na lak način radimo sa numeričkim vrednostima koje se zapisuju u polje stranog ključa. U polju tipa ComboBox će se prikazivati

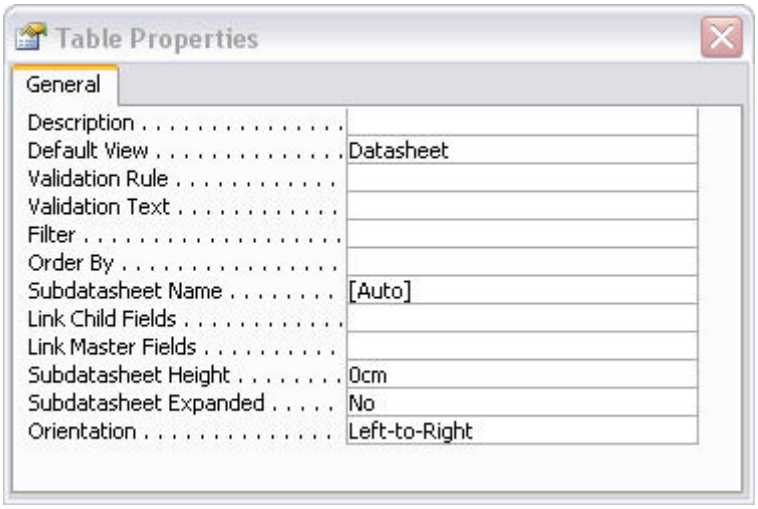


samo vrednosti koje se pronalaze pomoću SQL iskaza. Zbog toga je otkrivanje grešaka u aplikaciji otežano.

Osnovna prednost ove osobine jeste to što olakšava postupak pravljenja formulara jer u formularu automatski pravi polje tipa ComboBox kada se za polje tabele koristi osobina pronalaženja podataka. Ja smatram da je pravljenje polja tipa ComboBox u formularu toliko lako pa stoga ne vidim da je osobina lookup takva da može da uštedi vreme prilikom pravljenja formulara.

8.2 Rad sa svojstvima tabele

Pored svojstava polja, možemo zadati svojstva koja se odnose na tabelu. Da bismo došli do svojstava tabele, kliknite dugme Properties, koje se nalazi na paleti alata, kada se nalazimo u pogledu Design (za tabelu). Svojstva tabele vidite na *slici 3*. Svojstvo Description se uglavnom koristi radi dokumentovanja. Svojstvom Default View se određuje pogled u kojem se tabela prikazuje kada je korisnik prvi put otvori. Svojstvom Validation Rule se zadaju pravila provere koja se moraju zadovoljiti na nivou sloga, a ne na nivou polja. Na primer, ograničenje kreditiranja može zavisiti od stanja klijenta. U ovom slučaju, ono što se unese u jedno polje može zavisiti od vrednosti drugog polja. Ukoliko zadamo pravila provere na nivou tabele, nije važno u kom redosledu korisnik unosi podatke u polja. Pravila provere obezbeđuju primenjivanje zavisnosti između polja.



Slika 3: Prikazivanje svojstava tabele.

U svojstvu Validation Text se nalazi poruka koja se prikazuje korisniku kada naruši zadata pravila. Ukoliko svojstvu nije zadata vrednost, prikazuje se podrazumevana poruka.

Svojstvo Filter koristimo za izdvajanje podskupa slogova koji se prikazuju u tabeli, formularu ili upitu. Svojstva Filter i Order By se obično ne primenjuju kao svojstva tabele.

Svojstvom Subdatasheet Name se zadaje ime tabele koja se koristi za pronalaženje po vertikali. Ukoliko svojstvu zadate vrednost Auto, ova tabela se automatski pronalazi na onovu zavisnosti koje su uspostavljene u bazi podataka. Svojstva Link Child Fields i Link Master Fields se koriste za zadavanje polja, polja koja se koriste za uspostavljanje veze između tekuće tabele i tabele koja je naznačena svojstvom Subdatasheet Name. Ovim svojstvima ne treba da zadajemo vednost ukoliko je za svojstvo Subdatasheet Name zadata



vrednost Auto. Svojstvo Subdatasheet Height koristimo za zadavanje maksimalne visine podtabele, a svojstvo Subdatasheet Expanded korisimo da bismo odredili da li se podtabela automatski otvara.

8.2.1 Korišćenje indeksa radi poboljšanja performansi

Kao što je rečeno, indeksi mogu poboljšati performanse aplikacije. Indekse treba da pravimo nad poljima koje koristimo za sortiranje, grupisanje, spajanje ili za koja pravimo uslove, izuzev kada polja sadrže podatke koji se mnogo puta ponavljaju. Upiti će mnogo bolje raditi kada postoje indeksi, naročito kada su indeksi napravljeni nad poljima koja su deo uslova, nad poljima koja se koriste za uredivanje rezultata i kada se polja koriste za spajanje dve tabele za koje nije uspostavljena trajna veza već su povezane upitom. Zapravo, indekse treba uvek da pravimo za polja koja se koriste za spajanje tabela. Ukoliko korisnici upotrebljavaju okvir za dijalog Find, indeksi smanjuju vreme potrebno za pronalaženje padataka. Zapamtimo da su nedostaci indeksa prostor na hard disku koji zauzimaju i vreme koje je potrebno kada se slogovi dodaju, uklanjaju i ažuriraju. Uvek treba da testiramo performanse svojih aplikacija i verovatno ćemo uvideti da indeksi mogu da pomognu u mnogim situacijama.

Napomena

Kada između dve tabele uspostavimo zavisnost, za tabelu na strani "više" zavisnosti (to jest, za polje stranog ključa) automatski se pravi Indeks.

8.2.2 Accessove tabele i Internet

Microsoft je olakšao pravljenje aplikadja koje mogu koristiti za Internet stvarajući polje tipa Hyperlink i omogućavajući korisnicima da podatke tabele zapišu u HTML formatu. Polje tipa Hyperlink korisnicima omogućava da na lak način zapišu UNC ili URL adrese u tabele. Mogućnost zapisivanja podataka u HTML formatu nama i našim korisnicima omogućava prikazivanje podataka tabele na Internetu ili lokalnoj mreži.

8.2.3 Polje tipa Hyperlink

Korisnici pomoću polja tipa Hyperlink mogu u svaki slog tabele zapisati različite UNC ili URL adrese. Mada u polje možemo direktno upisati različite UNC ili URL adrese, mnogo je lakše te adrese uneti pomoću okvira za dijalog Insert Hyperlink. Pomoću ovog okvira za dijalog, korisnici grafički mogu pronalaziti adrese i podadrese, a adresa se automatski unosi kada zatvorimo okvir za dijalog. Da bismo otvorili okvir za dijalog Insert Hyperlik, kada se kursor nalazi u polju tipa Hyperlink, odaberimo Insert, Hyperlink ili kliknimo dugme Insert Hyperlink.

Tekstualno polje Text to Display se koristi za unošenje teksta koji će se prikazivati kada podatke tabele prikazuju u pogledu Datasheet, bilo u formularu ili izveštaju. Hiperlink može pokazivati na:



- Postojeću datoteku ili web stranicu
- Drugi objekat u tekućoj bazi podataka
- Novu stranicu za rad sa podacima
- E-mail aredresu

8.3 Ključ integriteta podataka

Zavisnost između dve tabele postoji kada je jedno ili više polja ključa jedne tabele upareno sa jednim ili više polja druge tabele. Polja u obe tabele obično imaju isto ime, tip podataka i veličinu. Zavisnosti su nusproizvod postupka normalizacije. Normalizacija podataka, jeste postupak eliminisanja dupliranih informacija iz sistema tako što se informacije dele u nekoliko tabela, od kojih svaka sadrži jedinstvenu vrednost (primarni ključ). Mada je normalizacija podataka veoma korisna, potrebno je da između tabela aplikacije uspostavimo veze tako da korisnici mogu videti podatke iz sistema kao jedan entitet. Pošto između tabela uspostavimo zavisnosti, možemo praviti upite, formulare, izveštaje i stranice za rad sa podacima na kojima se kombinuju informacije iz više tabela. Na ovaj način možemo iskoristiti sve dobrobiti normalizacije podataka, a da pri tom korisnici dobijaju sve informacije koje su im potrebne.

8.4 Ciljevi strukturiranja relacione baze podataka

Osnovni cilj strukture relacione baze podataka jeste da se, što je bolje moguće, napravi baza podataka koja predstavlja model nekog praktičnog sistema. To znači da praktični sistem rastavite u tabele i polja i da odredite kakvi su odnosi između tabela. Mada ovo može izgledati kao jednostavan posao, postupak prevođenja praktičnog sistema u tabele i polja ipak može biti izuzetno veliki.

Pravilno strukturirana baza podataka ima mnoge prednosti. Postupak unošenja, menjanja, uklanjanja i dobijanja podataka tabele je mnogo lakši kada je baza podataka pravilno strukturirana. Pored toga, izveštaje je lakše napraviti. Najvažnije od svega jeste da je bazu podataka lako menjati i održavati.

8.5 Pravila strukturiranja relacione baze podataka

Da bi se odslikao relacioni model, moraju se poštovati neka pravila. Ovim pravilima se određuje šta se zapisuje u tabelu i kakve veze postoje između tabela.

8.6 Tipovi zavisnosti između tabela

Izmedu tabela postoje tri tipa zavisnosti: jedan-prema-više, jedan-prema-jedan i više-prema-jedan. Obavezno je uspostavljanje odgovarajućeg tipa zavisnosti između tabela baze podataka. Odgovarajući tip zavisnosti izmedu tabela obezbeduje:

- Integritet podataka
- Optimalne performanse
- Laku upotrebu prilikom pravljenja objekata baze podataka



8.6.1 Zavisnost jedan-prema-više

Zavisnost jedan-prema-više je najvažniji tip zavisnosti između tabela. Kada je izmedu tabela uspostavljena zavisnost jedan-prema-više, onda jednom slogu tabele odgovara više slogova druge tabele. Da bismo implementirali zavisnost, polje po kojem se spajaju dve tabele na jednoj strani zavisnosti mora biti jedinstveno.

U gotovo svim slučajevima je polje po kojem se spajaju tabele primarni ključ jedne od tabela koje učestvuju u zavisnosti. Polje po kojem se spajaju tabele, a nalazi se na strani više zavisnosti, u toj tabeli je strani ključ.

8.6.2 Zavisnost jedan-prema-jedan

Kada je između tabela uspostavljena zavisnost jedan-prema-jedan, svakom slogu tabele sa jedne strane zavisnosti odgovara tačno jedan slog tabele sa druge strane zavisnosti. Ovakav tip zavisnosti nije uobičajen i koristi se samo u specijalnim slučajevima.

Ukoliko smo između tabela uspostavili zavisnost jedan-prema-jedan, onda je pre trebalo napraviti tabelu u kojoj su ta polja. Evo najčešćih razloga zbog kojih bi trebalo da uspostavljamo zavisnost jedan-prema-jedan:

- Broj polja u tabeli prelazi broj dozvoljenih polja u Accessovoj tabeli.
- Neka polja tabele se moraju bolje obezbediti od ostalih polja iste tabele.
- Nekoliko polja tabele su neophodna samo za podskup slogova iste tabele.

Maksimalan broj polja u Accessovoj tabeli jeste 255. Postoji veoma malo razloga zašto bi tabela imala više od 255 polja. Zapravo, pre nego što se približimo broju od 255 polja, trebalo bi dobro da proučimo strukturu baze podataka. U retkim slučajevima kada u tabeli ima smisla imati više od 255 polja, jednu tabelu možemo simulirati prebacivanjem nekih polja u drugu tabelu i uspostavljanjem zavisnosri jedan-prema-jedan između tih tabela.

Drugi razlog za razdvajenje podataka u dve tabele koji logički pripadaju istoj tabeli jeste bezbednost. Primer takvog razdvajanja jeste tabela koja sadrži informacije o zaposlenima. Neke informacije, kao što su ime zaposlenog, adresa, grad, država, poštanski broj, kućni telefonski broj i broj odeljenja u kome radi mogu videti mnogi korisnici baze podataka. Druga polja, uključujući datum zapošljavanja, platu, datum rodenja i platni nivo su poverljiva. U Accessu ne postoji bezbednost na nivou polja. Ovakvu bezbednost možemo simulirati korišćenjem specijalnog atributa upita koji se zove Run with Owner's Permissions.

Drugi način za postizanje bezbednosti jeste prebacivanje polja koje svi korisnici mogu da vide u jednu tabelu i prebacivanje poverljivih podataka u drugu tabelu. Samo specijalnom Admin korisniku (korisniku koji ima specijalne privilegije, ne onom čije je korisničko ime Admin) dozvoljavamo rad sa poverljivim podacima. Zatim, pomoću ActiveX Data Objects (ADO) koda prikazujemo podatke koji su poverljivi. To postižemo

tako što koristimo upit za koji je zadat atribut Run with Owner's Permissions, koji je napravljen na osnovu privilegija specijalnog Admin korisnika.

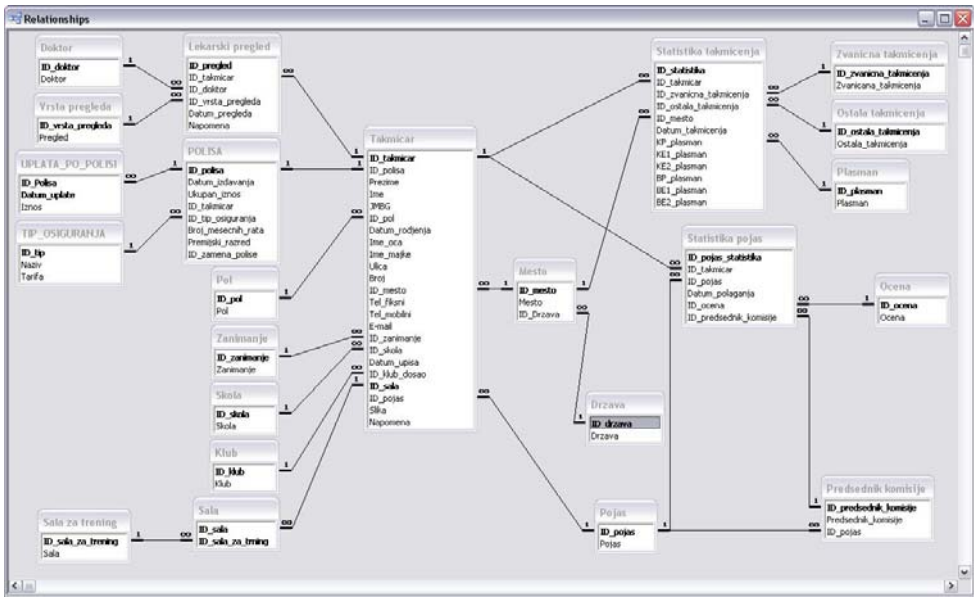
8.6.3 Zavisnost više-prema-više

Kada je izmedu tabela uspostavljena zavisnost više-prema-više, za slogove obe tabele postoje odgovarajud slogovi u drugoj tabeli. U Accessu ne možete direktno uspostaviti zavisnost više-prema-više. Ovakav tip zavisnosti moramo uspostaviti pomoću tabele koja se naziva tabela spajanja (junction table). Između tabele spajanja i druge dve tabele uspostavljamo zavisnost jedan-prema-više.

8.7 Uspostavljenje zavisnosti u Accessu

Prozor Relationships koristimo za uspostavljanje zavisnosti izmedu Accessovih tabela . Da bismo otvorili prozor Relationships, kliknite Relationships na paleri alata kada je aktivan prozor Database ili iz menija Tools odaberite Relationships. Ukoliko nismo uspostavili nijednu zavisnost, pojaviće se okvir za dijalog Show Table. Okvir za dijalog Show Table nam omogućava da u prozor Relationships dodamo tabele.

U prozoru Relationships možemo videti tip zavisnosti koji je uspostavljen za svaku tabelu. Sve uspostavljene zavisnosti jedan-prema-više i jedan-prema-jedan su predstavljene linjom. Ukoliko za tabele za koje je uspostavjena zavisnost jedan-prema-više primenite pravila referencijalnog integriteta, pored linije kojom je predstavljena zavisnost, na strani jedan se prikazuje broj 1, a na strani vise se prikazuje simbol za beskonačnost (∞). Pored linije kojom je predstavljena zavisnost jedan-prema-jedan na oba kraja se prikazuje broj 1.



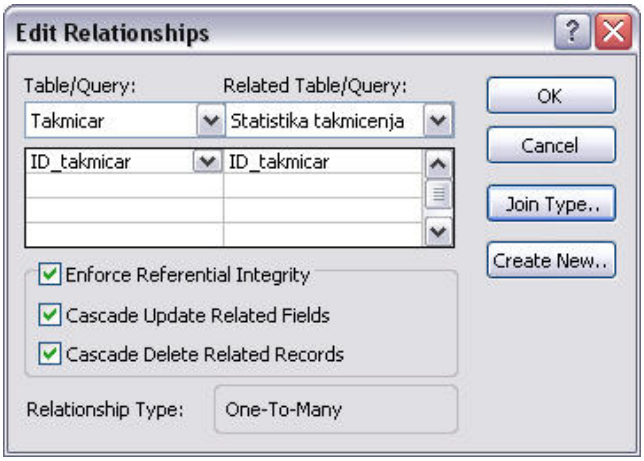
Slika 4: Prozor Relationships nam omogućava da prikažemo, napravimo, izmenimo i uklonimo zavisnosti između tabela.



8.8 Uspostavljanje zavisnosti između dve tabele

Da bismo uspostavili zavisnost između dve tabele, pratimo sledeće korake:

- Otvorimo prozor Relationships.
- Ukoliko je to prvo otvaranje prozora Relationships u bazi podataka, prikazaće se okvir za dijalog Show Table. Odaberimo jednu po jednu tabelu između kojih želimo da uspostavimo zavisnosti i kliknimo Add.
- Ukoliko smo u tekućoj bazi već uspostavili zavisnosti, prikazaće se prozor Relationships. Ukoliko se u prozoru ne nalaze tabele između kojih želimo da uspostavimo zavisnosti, kliknimo dugme Show Table ili iz menija Relationships odaberimo Show Table. Da bismo tabelu dodali u prozor Relationships, odaberimo tabelu i kliknimo Add. Ovaj postupak ponovimo za svaku tabelu koju želimo da dodate u prozor Relationships. Da bismo odjednom odabrali više tabela, pritisnimo taster Shift i pri tom odaberimo uzastopne tabele iz spiska tabala ili pritisnimo taster Ctrl i odaberimo tabele koje nisu uzastopne u spisku.
- Kliknimo i prevucimo polje iz jedne tabele do odgovarajućeg polja druge tabele. Prikazaće se okvir za dijalog Edit Relationships.
- Odlučimo da li ćemo uspostaviti pravila referencijalnog integriteta i da li ćemo kaskadno ažurirati polja ili ćemo polja kaskadno uklanjati tako što ćemo potvrditi odgovarajuća polja.
- Kliknimo Create.



Slika 5: Okvir za dijalog Edit Relationship nam omogućava da prikažemo i izmenimo zavisnosti izrmeđu tabela baze podataka.

8.8.1 Menjanje postojećih zavisnosti

Menjanje postojeće zavisnosti je veoma lako. U Accessu možemo ukloniti zavisnost ili promeniti njenu prirodu. Da bismo trajno uklonili zavisnost koja je uspostvaljena između dve tabele, uradimo sledeće:

- Dok je aktivan prozor Database, na paleti alata kliknimo Relationships.
- Kliknimo liniju koja spaja dve tabele i predstavlja zavisnost koju želimo da uklonimo.
- Pritisnimo taster Delete. Access će zatražiti potvrdu da zaista želimo da uklonimo zavisnost. Kliknimo Yes.



- Često ćemo hteti da promenite prirodu zavisnosti, a ne da je uklonite. Da bismo izmenili zavisnosti, uradite sledeće:
- Dok je aktivan prozor Database, na paleti alata kliknimo Relationships.
- Dva puta kliknimo liniju koja spaja dve tabele i predstavlja zavisnost koju želimo da izmenimo.
- Napravimo neophodne izmene.
- Kliknimo OK. Sva pravila koja se tiču uspostavljanja zavisnosti će biti primenjena.

8.8.2 Uspostavljanje referencijalnog integriteta

Kao što ste saznali, uspostavljanje zavisnosti izmedu tabela je veoma lako. Uspostavljanje odgovarajuće vrste zavisnosti je nešto teži zadatak. Kada pokušate da uspostavite zavisnost između dve tabele, Access donosi nekoliko odluka na osnovu nekoliko unapred zadatih faktora:

- Access uspostavlja zavisnost jedan-prema-više ukoliko je jedno od povezanih polja primarni ključ ili jedinstveni indeks.
- Access uspostavlja zavisnost jedan-prema-jedan ukoliko su oba polja koja su povezana deo primarnog ključa ili imaju jedinstveni indeks.
- Access pravi neodređenu zavisnost ukoliko nijedno polje nije deo primarnog ključa i nijedno nema jedinstveni indeks. U ovom slučaju ne možete uspostaviti referencijalni integritet.

Kao što je objašnjeno ranije u ovom poglavlju, *referencijalni integritet* se sastoji od niza pravila koje primenjuje Jet Engine kako bi obezbedio pravilno održavanje zavisnosti između tabela. Na najnižem nivou, pravila referencijalnog integriteta sprečavaju postojanje slogova-siročića u tabeli koja se nalazi na strani vise zavisnosti jedan-prema-više. Nakon uspostavljanja zavisnosti između tabela tblCustomers i tblOrders, na primer, sve porudžbine u tabeli tblOrders moraju imati odgovarajućeg klijenta u tabeli tblCustomers. Pre nego što izmedu dve tabele uspostavite referencijalni integritet, moraju se ispuniti sleded uslovi:

- Odgovarajuće polje na jednoj strani zavisnosti mora biti deo primarnog ključa ili mora imati jednistveni indeks.
- Odgovarajuća polja moraju biti istog tipa podataka (zbog uspostavljanja veze, polju tipa AutoNumber odgovara polje tipa Long Integer). Sa izuzetkom polja tipa Text, moraju biti i iste veličine. Polja tipa Number sa obe strane zavisnosti moraju imati istu veličinu (na primer, Long Integer).
- Obe tabele se moraju nalaziti u istoj bazi podataka.
- Obe tabele se moraju nalaziti u istom Accessovom formatu datoteke (MDB) (tabele ne mogu biti spoljašnje tabele).
- Tabela koja sadrži dve tabele mora biti otvorena.
- Postojeci podaci u tabelama ne smeju narušavati bilo koje pravilo referencijalnog integriteta. Za sve porudžbine iz tabele tblOrders mora postojati klijent u tabeli tblCustomers, na primer.



Upozorenje

Iako polja tipa Text koja su deo zavisnosti ne moraju biti iste veličine, mudro je učiniti da budu iste veličine. U suprotnom, umanjićemo performanse i uvećaćemo šanse za dobijanje nepredviđenih rezultata prilikom pravljenja upita u kojima se koriste te dve tabele.

Pošto između dve tabele uspostavimo referencijalni integritet, Jet Engine primenjuje sledeća pravila:

- Ne možete uneti vrednost u strani ključ povezane tabele koja ne postoji u primarnom ključu primarne tabele. Na primer, ne možete uneti vrednost u polje CustomerlD tabele tblOrders ukoliko ta vrednost ne postoji u polju CustomerlD tabele tblCustomers.
- Iz primarne tabele ne možete ukloniti slog ukoliko odgovarajući slogovi postoje u povezanoj tabeli. Na primer, ne možete^ ukloniti slog o klijentu iz tabele tblCustomers ukoliko za taj slog postoje odgovarajući slogovi u tabeli tblOrders (ukoliko postoje slogovi koji imaju istu vrednost u polju CustomerlD).
- Ne možete promeniti vrednosti primarnog ključa na jednoj strani zavisnosti ukoliko u povezanoj tabeli postoje odgovarajući slogovi. Na primer, ne možete promeniti vrednost u polju CustomerlD tabele tblCustomers ukoliko u tabeli tblOrders postoji odgovarajući slog.

Ukoliko pokušamo da narušimo bilo koje od prethodna tri pravila, a između tabela smo primenili pravila referencijalnog integriteta, Access prikazuje odgovarajuću poruku o grešci.

Osnovno ponašanje Jet Enginea jeste zabrana uklanjanja roditeljskih slogova za koje postoje dete-slogovi i zabrana promene vrednosti primarnog ključa roditeljskog sloga kada za njega postoje dete-slogovi. Ova pravila možemo zaobići koristeći dva polja za potvrdu koja se nalaze u okviru za dijalog Relationships prilikom uspostavljanja i menjanja zavisnosti.

8.8.3 Kaskadno ažuriranje povezanih polja

Opcija Cascade Update Related Fields se može koristiti samo ukoliko ste između tabela uspostavili pravila referencijalnog integriteta. Kada je ova opcija odabrana, korisnik može promeniti vrednost primarnog ključa sloga koji se nalazi strani jedan uspostavljene zavisnosti.

Kada korisnik pokuša da izmeni sadržaj polja na strani jedan zavisnosti po kojem je uspostavljena veza između dve tabele, Jet Engine prosleduje izmenu do polja stranog ključa na strani vise uspostavljene zavisnosti. Ova osobina je korisna ukoliko se polje primarnog ključa može menjati. Ukoliko korisnik izmeni broj porudžbenice roditeljskog sloga, onda bi ta izmena trebalo da se prosledi u odgovarajuće slogove tabele u kojoj se čuvaju detalji o porudžbenici.



Kada je polje na strani jedan uspostvaljene zavisnosti tipa AutoNumber, nije neophodno odabrati opciju Cascade Update Related Fields. Vrednost polja tipa AutoNumber nikada ne možete izmeniti. Opcija Cascade Update Related Fields nema nikakav uticaj na polja tipa AutoNumber.

Veoma je lako u sistemu slučajno napraviti prazninu. Ukoliko između tabela uspostavite zavisnost jedan-prema-više, ali zaboravite da svojstvu Required polja stranog ključa zadate vrednost Yes, onda ste dozvolili unošenje slogova za koje ne postoji roditeljski slog. Na slici 3.8 je ilustrovan ovaj problem. U tabelu tbIOrders sam dodala porudžbinu za koju u tabeli tbICustomers ne postoji klijent. To je slog za koji ne postoji roditeljski slog jer u tabeli tbICustomers nema klijenata čiji je identifikator vrednost Null. Da biste eliminisali problem, svojstvu Required polja stranog ključa zadajte vrednost Yes.

8.8.4 Kaskadno uklanjanje povezanih slogova

Opcija Cascade Delete Related Records se može koristiti samo ukoliko smo između tabela uspostavili pravila referencijalnog integriteta. Kada je ova opcija odabrana, korisnik može ukloniti slog na strani jedan uspostavljene zavisnosti jedan-prema-više, čak i kada u tabeli na strani više te zavisnosti postoje odgovarajući slogovi. Korisnik može ukloniti podatke o klijentu čak i kada postoje porudžbine. Jet Engine održava referencijalni integritet između tabela jer automatski uklanja sve slogove iz dete-tabele koji se odnose na roditeljski slog.

Ukoliko pokušamo da uklonimo slog iz tabele koja se nalazi na strani jedan uspostavljene zavisnosti jedan-prema-više, a u tabeli na strani više te iste zavisnosti ne postoje odgovarajući slogovi, prikazaće se neuobičajena poruka.

Sa druge strane, ukoliko pokušamo da uklonimo slog iz tabele koja se nalazi na strani jedan zavisnosti jedan-prema-više, a u dete-tabeli postoje odgovarajući slogovi, Access nas upozorava da ćemo iz roditeljske tabele ukloniti slog i da će svi odgovarajući slogovi iz dete-tabele takođe biti uklonjeni.

Savet

Opciju Cascade Delete Related Records nije uvek pogodno koristiti. To je sigurno izuzetna osobina, ali je treba mudro koristiti.

Kada je odabrana opcija Cascade Update, možemo ažurirati vrednost primamog ključa slogova za koje postoje dete-slogovi. Kada je odabrana opcija Cascade Delete, možemo ukloniti roditeljski slog za koji postoje dete-slogovi. Ova vežba ilustruje kako da koristimo opcije Cascade Update i Cascade Delete.



8.8.5 Dobrobiti zavisnosti

Osnovna dobrobit zavisnosti jeste integritet podataka koji obezbeđuje. Ukoliko se zavisnosti ne uspostave, korisnici slobodno mogu u dete-tabele unositi slogove, a da pri tom ne moraju da unose neophodne roditeljske informacije. Pošto se uspostavi referencijalni integritet, možete, po potrebi, korisiti opcije **Cascade Update Related Fields** i **Cascade Delete Related Records**, pa nećemo morati da pišemo mnogo koda radi održavanja integriteta podataka u bazi podataka. Većina sistema za upravljanje relacionim bazama podataka zahtevaju pisanje koda za uklanjanje povezanih slogova kada korisnik uklanja roditeljski slog ili za ažuriranje stranog ključa povezanih slogova kada korisnik menja primarni ključ roditeljskog sloga. Ukoliko potvrdite polja Cascade Update i Cascade Delete, nećemo morati da napišemo niti jedan red koda kojim bismo obavili ove poslove.

Access automatski uvodi zavisnosti u upite. To znači da svaki put kada pravimo nov upit, on automatski uspostavlja zavisnosti između tabela koje se koriste u upitu, i to na osnovu zavisnosti koje smo zadali u prozoru Relationships. Osim toga, svaki put kada pravite formular ili izveštaj, Access koristi zavisnosti izmedu tabela koje se koriste u formularu ili izveštaju kako bi nam pomogao prilikom postupka pravljenja. Bilo da uklanjamo ili ažuriramo podatke koristeći tabelarni prikaz ili formular, sva pravila referencijalnog integriteta se automatski primenjuju, čak i kada zavisnosti uspostavimo nakon što smo formular napravili.

8.8.6 Proučavanje indeksa i zavisnosti

Polje pomoću kojeg su spojene tabele na strani jedan zavisnosti jedan-prema-više mora biti polje primarnog ključa ili mora imati jedinstven indeks kako bi Jet Engine mogao da održava referencijalni integritet. Ukoliko indeks na strani jedan uspostavljene zavisnosti nije jedinstven, ne postoji način da znamo kom roditeljskom slogu pripradaju dete-slogovi.

U Accessu 2003 nije neophodno napraviti indeks za polje na strani više uspostavljene zavisnosti. Access 2003 će za nas napraviti interni indeks. Ukoliko ipak na strani više zavisnosti jedan-prema-više napravimo indeks, moramo za taj indeks zadati vrednost Yes (Duplicates OK). U suprotnom ćemo dobiti zavisnost jedan-prema-jedan, a ne zavisnost jedan-prema-više.

8.9 Istorijat relacionih baza podataka

Dr E.E Kod (E.F. Codd) je 1969. godine, dok je još uvek radio u IBM-u, uveo formalnu strukturu relacione baze podataka. Relaciona teorija, koja se zasniva na teoriji skupova, odnosi se na baze podataka i aplikacije za baze podataka. Dr Kod je uspostavio 12 pravila pomoću kojih se određuje koliko se dobro aplikacija i njeni podaci uklapaju u relacioni model. Od vremena kada je Dr Kod smislio tih 12 pravila, njihov broj se popeo na stotine. (Potrebno je da naučimo samo nekoliko!)



8.10 Pravila za jedinstvenost i ključeve

Tabele se sastoje iz redova i kolona. Da bi odslikala relacioni model, svaka tabela mora sadržati jednistveni identifikator. Bez jedinstvenog identifikatora je nemoguće programski jedinstveno adresirati red (slog). Jedinstvenost u tabeli se garantuje stvaranjem primarnog ključa, a to je jedna kolona ili skup kolona koje jedinstveno identifikuju red tabele.

Svaka kolona ili skup kolona tabele koje sadrže jedinstvene vrednosti se smatraju mogućim ključem (candidate key). Jedan od mogućih ključeva postaje primarni ključ. Ostali mogući ključevi postaju alternativni ključevi (alternate key). Primarni ključ koji je napravljen nad jednom kolonom se naziva jednostavan ključ (simple key). Primarni ključ koji je napravljen nad više kolona se naziva složeni ključ (composite key).

Obično je dobro da je primarni ključ:

- Minimalan (ima što je manje moguće kolona)
- Stabilan (da se retko menja)
- Jednostavan (poznat korisniku)

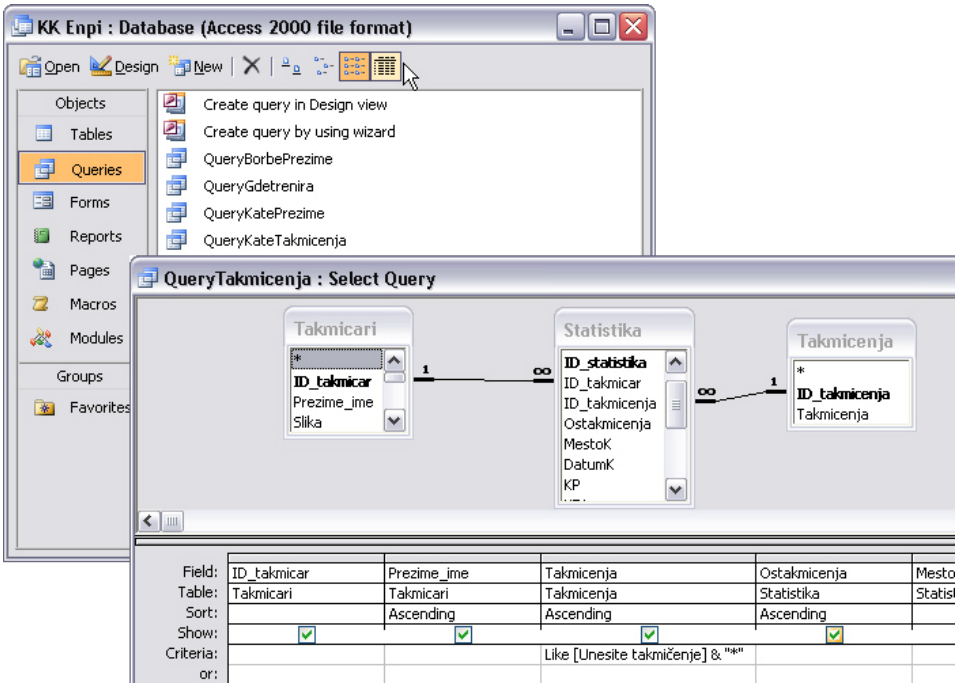
Poštovanje ovih pravila značajno poboljšava performanse i olakšava održavanje aplikacije baze podataka, naročito ako radimo sa velikom količinom podataka.

8.10.1 Pravila za strane ključeve i domene

Strani ključ (foreign key) u jednoj tabeli jeste polje koje je povezano sa primarnim ključem druge tabele.

Domen (domain) jeste skup vrednosti koje se mogu zapisati u kolonu. Jednostavan primer domena jeste raspon datuma zapošljavanja zaposlenih.

9 UPITI



Slika 6: Izgled prozora za kreiranje Querijaodnosno Upita

Upiti u Accessu su veoma moćni i svestrani. Upiti odabiranja nam omogućavaju da prikažemo, sažmemo i obavimo izračunavanja sa podacima koji se nalaze u tabelama. Akcioni upiti nam omogućavaju da unesemo i ažuriramo podatke u tabelu i izbrišemo ih iz tabele.

Da bismo izvršili upit, iz spiska Objects odaberimo Queries, dva puta kliknite upit koji želimo da izvršimo, ili označimo upit koji želimo da izvršimo, a zatim kliknimo Open. Pošto izvršimo odabrani upit, u tabelarnom obliku će se prikazati sva polja koja su zadata upitom i svi slogovi koji zadovoljavaju kriterijum upita. Pošto pokrenemo akcioni upit, Access obavlja zadatu akciju, recimo, pravljenje nove tabele ili dodavanje podataka u postojeću tabelu. U principu, podatke koje dobijamo kao rezultat upita možemo ažurirati jer je rezultat upita zapravo dinamički skup slogova, na osnovu podataka tabele.

Znači, upiti su posebni pregledi kojima se postavlja neki upit nad tabelom i koji izdvajaju podatke za pregled na ekranu ili štampanje putem izveštaja. Npr. ukoliko želimo pregled svih takmičenja održanih posle meseca januara u kriterijum za upit ćemo upisati **>31.01.2005**.

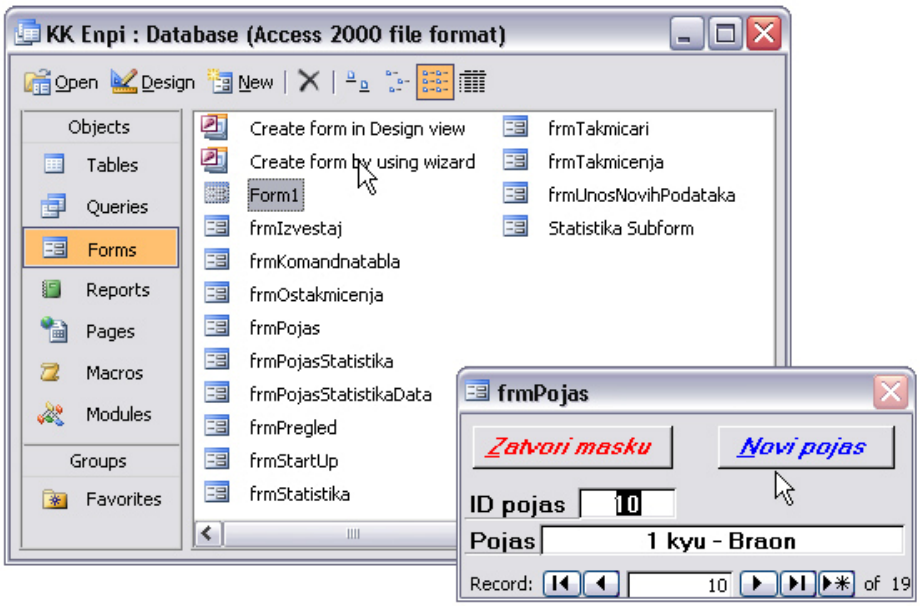
Upiti mogu filtirati podatke ne samo iz jedne tabele već se to može činiti iz više tabela ili čak iz jednog ili više drugih upita. Takođe mogu istovremeno filtrirati podatke po više kriterijuma.

Pošto pokrenemo upit Employee Sales by Country, pojaviće se tabelami prikaz podataka koji obuhvata sva polja zadata upitom i sve slogove koji zadovoljavaju kriterijum upita.

Pošto zapišemo upit, samo njegovu definiciju, plan ili svojstva formatiranja, i tabelami prikaz su zapravo zapisani u bazi podataka. Access nam za strukturiranje uputa nudi intuitivni alat. Na slici 6 vidite prozor Query Design. Da bismo otvorili ovaj prozor, u prozoru Database iz spiska Objects odaberimo Queries, zatim odaberimo upit koji želimo da izmenimo, pa kliknimo Design. U toku izvrsavanja upit traži kriterijum izvršavanja i pomoću njega određuje koje slogove će prikazati u rezultatu.

Sem izdvajanja podataka postoje i posebni upiti koji mogu kreirati tabelu koristeći drugu tabelu ili više drugih tabela nad kojom je postavljen upit, odnosno mogu brisati podatke iz postojeće tabele takođe po nekom kriterijumu. Postoje i drugi: **Update** upit (proračunava podatke i rezultat upisuje u polje tabele, npr. **ukupno=cena*količina** - upisuje rezultat u polje **ukupno**), **Union** upit, **Crosstab** upit (proračun nad prividnom matricom) i **Append** upit koji dodaje slogove u postojeću tabelu.

Forme: Načini prikazivanja, menjanja i dodavanja podataka (formulari, obrasci ili maske za unos i pregled)



Slika 7:
Izgled prozora za kreiranje Formi

Iako u pogledu Datasheet tabele možemo unositi i menjati podatke, ne možemo dovoljno dobro kontrolisati akcije korisnika, niti možemo učiniti mnogo toga kako bismo olakšali postupak unošenja podataka. Zbog toga se koriste formulari. Accessovi formulari imaju mnogo osobina i veoma su moćni i fleksibilni.



Da bismo prikazali bilo koji formular, iz spiska Objects odaberimo Forms. Zatim dva puta kliknimo formular koji želimo da prikažemo, ili obeležimo formular koji želimo da prikažemo tako što ćemo na njega kliknuti pa zatim kliknimo Open.

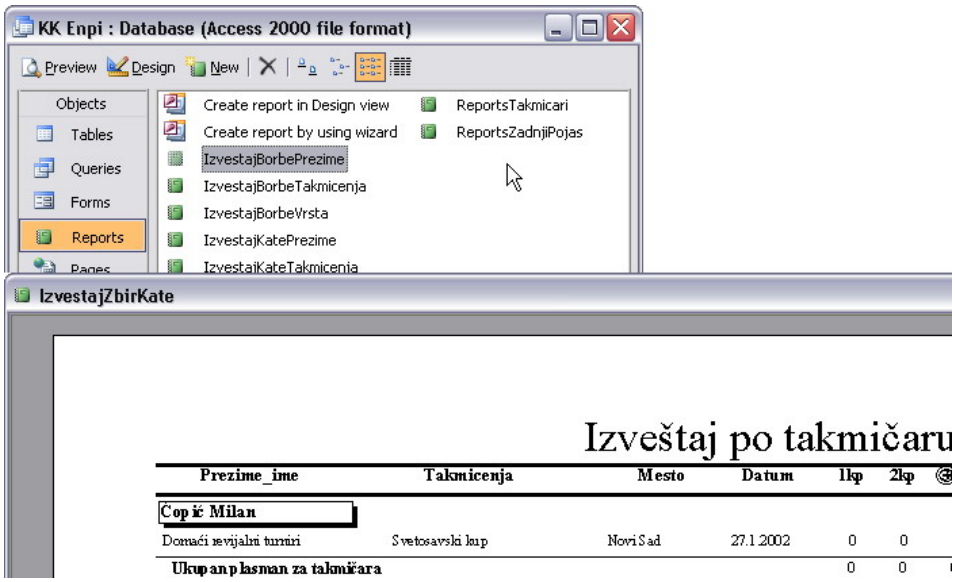
Kao što to možemo da uradimo za tabele i upite, i formulare možemo prikazati u pogledu Design. Da bismo videli strukturu formulara, iz spiska Obejcts odaberimo ikonu Forms, odaberimo formular čiju strukturu želimo da promenimo, pa zatim kliknimo Design.

Forma je samo "pogled" na tabelu u čoveku prihvatljivijem izgledu. Nad formiranom tabelom koja sadrži podatke o nekom članu formiramo masku za unos koja ima izgled papirnog obrasca - kartice, te manipulant ne mora da poznaje rad sa Accessom, nego samo koristi elektronske obrasce (maske) i unosi podatke putem tastature, baš kao što je to ranije činio olovkom.

Forma za unos podataka može sadržati slike, komandne tastere (pritiskom na njih se aktivira neka komanda kao što je štampanje trenutne kartice ili prelaz na drugu formu), može sadržati logička polja predstavljena putem prekidača, može sadržati podformu itd.

Forma ne mora da sadrži niti jedno polje za unos ili prikaz podataka. Ona jednostavno može biti pozdravni ekran vaše aplikacije sa prigodnim tekstom i slikom, koje se sama posle isteka nekog vremenskog intervala zatvara. Takve forme mogu uzeti oblik pozdravne slike koja se pojavljuje prilikom starta nekog većeg programa, kao što se to događa u programima paketa Microsoft Office.

10 IZVEŠTAJI: Pretvaranje podataka u informacije



Slika 8: Izgled prozora za kreiranjeIzveštaja

Izveštaji su elektronski oblik klasičnih papirnih obrazaca koje možemo gledati na ekranu ili štampati na papir. Podaci prikazani u izveštaju mogu biti izvedeni iz tabela ili iz upita (filtrirani podaci). Dizajniranje izveštaja je veoma slično dizajniranju formi.

Kada kreiramo izveštaj za koji očekujemo da će imati više strana, u nekom od zaglavlja strane dodelimo i svojstvo **[Pages]** koje daje ukupan broj strana. Izveštaj u čijem dnu piše, recimo, **"Strana 6 od 75"** ne samo da daje korisnu informaciju, već svakako daje utisak ozbiljnog dokumenta.

Dok formiramo izgled lista izveštaja, često prelazimo iz Design prozora u Print Preview, kako bismo stekli prvi utisak o nekom izvedenom koraku. Ako smo definisali svojstvo **[Pages]**, svaki put će se ova vrednost ponovo izračunavati prolaskom kroz celu bazu i nepotrebno nam oduzimati vreme. Možda i nije problem ako je reč o izveštaju od nekoliko strana, ali šta ako je to nekoliko stotina strana? Može se desiti da čekamo i po nekoliko minuta (!) samo da bismo bacili brzi pogled na izgled stranice izveštaja: rad će se pretvoriti u agoniju čekanja, a efikasnost rada će jako opasti.

Zato, ako imamo nameru da koristimo ovo svojstvo, nećemo ga definisati pre samog kraja posla. U međuvremenu u zaglavlje stavimo neki običan tekst (recimo, **"OvdeBrojStrana"**), koji će nas podsetiti na taj detalj. U slučaju rada sa velikom bazom, razlika u vremenu dobijanja rezultata u prozoru Print Preview će biti drastična.



11 MAKROI I VBA

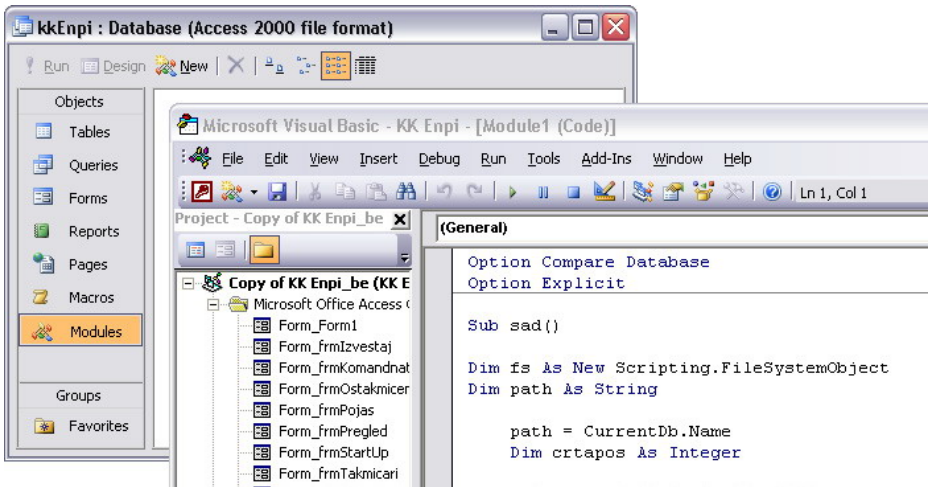
Makroi i VBA predstavljaju dva načina za programiranje baze podataka pisanjem programa. Često se javlja potreba za formiranjem funkcija koje ne postoje u Accessu, ili se grupe funkcija moraju vezati u jedan sklop, te se za razvijanje i automatizovanje baze podataka mora pristupiti pisanju programa. Makroima se iz konačnog skupa funkcija i naredbi automatizuje rad baze; međutim, njihovo korišćenje se u novijim aplikacijama izbegava, jer se puna sloboda i funkcionalnost ostvaruju tek u VBA. Makroi, zapravo, postoje isključivo zbog kombatibilnosti sa prethodnim verzijama Accessa. O programiranju ćemo mnogo više i detaljnije govoriti kasnije, jer predstavlja najvažniji deo funkcionalne aplikacije.

11.1 Makroi: Način za automatizovanje sistema

Makroi u Accessu nisu kao makroi u drugim članovima paketa Office. Ne možemo ih snimiti, kao što to možemo uraditi npr. u Excelu, a Access ih ne zapisuje kao kod u programskom jeziku Visual Basic for Applications (VBA). Pomoću Accessovih makroa možemo obaviti većinu poslova koje možemo obaviti pomoću tastature, menija i paleta alata. Makroi nam omogućavaju da u aplikaciju ugradimo logiku. Uopšte, radije koristimo VBA kod koji se nalazi u modulima, a ne u makroima, za obavljanje poslova koje aplikacija mora da obavi. To je zato što su moduli VBA koda fleksibilniji i moćniji od makroa. Iako su se u ranijim verzijama Accessa neki poslovi mogli obaviti samo pomoću makroa, programeri ih danas veoma malo koriste. Stoga se u ovom radu neću baviti pravljenjem aplikacija pomoću makroa.

Da bismo pokrenuli makro, iz spiska Objects odaberite Macros, kliknimo makro koji želimo da pokrenemo, pa zatim kliknimo Run. Access će zatim izvršiti akcije koje su navedene u makrou. Da bismo videli strukturu makroa, iz spiska Objects odaberimo Macro, obeležimo makro koji želimo da izmenimo, a zatim kliknimo design kako bismo ga otvorili u prozoru Macro.

11.2 Moduli: Osnova postupka pravljenja aplikacije



Slika 9: Izgled prozora za kreiranje Modula

Moduli, osnova svake aplikacije, nam omogućavaju da napravimo biblioteke funkcija koje možemo koristiti u aplikacijama. U modulima koje pravimo obično smeštamo podrutine i funkcije. Rezultat rada funkcije je uvek vrednost; podrutine kao rezultat rada ne daju vrednost.

Korišćenjem modula koda možete uraditi sledeće:

- Obraditi greške
- Deklarisati i koristiti promenljive
- Raditi sa skupovima slogova
- Pozivati funkcije Windowsovog API-ja i drugih biblioteka
- Praviti i menjati sistemske objekte, kao što su tabele i upiti
- Obavljati obradu pomoću transakcija
- Obavljati mnoge funkcije koje ne možemo obavljati pomoću makroa
- Testirati i debagovati složene procese
- Praviti biblioteku baza podataka

To su samo neki od poslova koje možemo obavljati pomoću modula. Da bismo videli strukturu postojećeg modula, u spisku Objects kliknimo Modules, odaberimo modul koji želimo da izmenimo, pa kliknimo Module kako bismo otvorili okvir za dijalog Module Design (videti sliku 9).



12 NORMALIZACIJA I NORMALNE FORME

Neke od najtežih odluka koje programer mora doneti jesu koje tabele treba napraviti i koja polja će te tabele sadržati, kao i kakve veze između tabela da uspostavi. Normaiizacija (normalization) jeste postupak primenjivanja niza pravila kojima se obezbeđuje da baza podataka ostvari optimalnu strukturu. Normalne forme (normal form) su niz tih pravila. Svakom narednom normalnom formom se dobija bolja struktura baze podataka nego što je dobijena prethodnom normalnom formom. Mada postoji nekoliko nivoa normalnih formi, u opštem slučaju je dovoljno zadovoljiti prve tri normalne forme.

12.1 Prva normalna forma

Da bismo zadovoljili prvu normalnu formu, sve kolone tabele moraju biti atomične. To, na primer, znači da u istom polju ne možemo zapisivari lično ime i prezime. Razlog za postojanje ovog pravila jeste to što može biti veoma teško raditi i dobijati podatke kada se u jednom polju zapisuje više vrednosti. Ako za primer uzmemo puno ime osobe, nemoguće je sortirati podatke prema ličnom imenu ili prezime ukoliko se obe vrednosri nalaze u istom polju. Osim toga, korisnik ili mi moramo obaviti dodatni korak ukoliko iz polja želimo da izdvojimo lično ime ili prezime.

Sledeći zahtev koji tabela mora zadovoljiti da bi bila u prvoj normalnoj formi jeste da ne sme da sadrži vrednosti koje se ponavljaju.

12.2 Druga normalna forma

Da bi tabela bila u drugoj normalnoj formi, sva polja koja nisu deo ključa moraju u potpunosti zavisiti od primarnog ključa. Drugim rečima, u svakoj tabeli se moraju zapisivati podaci o samo jednom subjektu.

Postupak razdvajanja podataka u dve tabele se naziva dekompozicija. Kada se prilikom dekompozicije ne gube podaci, takva dekompozicija se naziva dekompozicija bez gubljenja podataka (non-loss decomposition). Pošto podatke razdvojimo u dve tabele, podatke lako možemo upariti spajanjem tabela pomoću upita.

12.3 Treća normalna forma

Da bismo postigli treću normalnu formu, tabela mora da zadovolji sve zahteve prve i druge normalne forme, a sva polja koja nisu deo ključa ne smeju da zavise jedno od drugog. To znači da moramo eliminisati sva izračunavanja i da podatke moramo razdvojiti u tabele koje se pretražuju (lookup tabele).

Primer zapisvanja vrednosti koja se izračunava u tabeli jeste ukupna vrednost naručenih proizvoda koja se dobija množenjem vrednosti jednog proizvoda i naručene količine. Umesto da rezultat ovog izračunavanja zapisujemo u tabelu, treba da napravimo upit pomoću kojeg se vrednost izračunava ili da izračunavanje obavimo u kontroli koja se nalazi u formularu ili izveštaju.



12.4 Denormalizacija - namerno kršenje pravila

Iako je cilj programera noramlizacija, u mnogim slučajevima ima smisla odstupiti od normalnih formi. Ovakav postupak nazivamo denormalizacija (denormalizatiori). Osnovni razlog za denormalizaciju jeste poboljšanje performansi.

Primer kada je denormalizacija korisna jeste kada imamo tabelu otvorenih faktura i tabelu računa. Nije praktično po potrebi sumirati račune za klijenta. Umesto toga račune možemo sumirati u posebnoj tabeli tako da ih lako možemo dobiti kada su nam potrebni. Mada se na ovaj način poboljšavaju performanse, nedostatak je to što svaki put kada zamenimo otvorenu fakturu, moramo ažurirati sumarnu tabelu. To nedvosmisleno dovodi do odluke da li su nam bitnije performanse ili lakoća održavanja. Sami moramo odludti šta ćemo žrtvovati.

12.5 Pravila integriteta

Mada pravila integriteta nisu sastavni deo normalnih formi, ova pravila su svakako sastavni deo postupka strukturiranja baze podataka. Pravila integriteta se dele u dve kategorije. To su opšta pravila i pravila koja su specifična za bazu podataka.

12.6 Opšta pravila

Dve vrste opštih pravila su pravila referencijalnog integriteta i pravila integriteta entiteta. Pravila referencijalnog integriteta nalažu da u bazi podataka ne smeju postojati neuparene vrednosti stranog ključa. To znači da:

- U tabelu se ne mogu uneti redovi ukoliko za njih ne postoji odgovarajud red u drugoj tabeli. Drugim rečima, ne može se uneti porudžbina za klijenta koji ne postoji.

- Vrednost primarnog ključa se ne može menjati ukoliko se ta vrednost korisri kao strani ključ u drugoj tabeli.

- Iz tabele se ne može ukloniti red ukoliko u drugoj tabeli postoje redovi u kojima je ta vrednost deo stranog ključa.

Pravila integriteta entiteta nalažu da vrednost primarnog ključa ne može biti vrednost Null. Ovo pravilo se ne odnosi samo na primarne ključeve koji su napravljeni nad jednom kolonom, već i na primarne ključeve koji su napravljeni nad više kolona. Zapravo, u primarnom ključu koji je napravljen nad više kolona nijedno polje ne može sadržati vrednost Null. To ima smisla jer, kada bi primarni ključ mogao imati vrednost Null, on se više ne bi mogao koristiti kao jedinstveni identifikator reda. Na sreću, Jet Engine (Accessov mehanizam baze podataka) ne dozvoljava da polje primarnog ključa dobije vrednost Null.



13 MODELOVANJE PROCESA

U kompleksnom poslovnom svetu današnjice, koji je u fazi neprestanih promena, najvažniji proces predstavlja zadovoljenje potreba i želja kupaca – potrošača. Bez obzira da li se govori o malim ili velikim preduzećima i da li je krajnji produkt poslovanja materijalno dobro ili usluga, kvalitet dovodi do primarnog mesta na listi uspešnosti. Korišćenje modelovanja poslovnog procesa omogućuje bolje razumevanje mnogobrojnih interakcija, kako sa spoljnim svetom, tako i unutar organizacije, što dovodi do njegovog poboljšanja. Modelovanje je jedan od najefikasnijih načina za razumevanje i unapređenje poslovnih pravila. U modelu se manje značajni detalji zanemaruju, a oni od veće važnosti ističu, kako bi se kompleksnost sistema koji se posmatra smanjila, a važni elementi bolje uočili. Grafička prezentacija u vidu pravougaonika i strelica obezbeđuje slikovito predstavljanje poslovnog sistema, kako pojedinih delova od kojih se sastoji, tako i njegove celine. Modelovanjem je omogućeno da se sistem od interesa razloži do najsitnijih elemanata, tako da je moguće analizirati i najskrivenije detalje, razumeti ih i što je možda najvažnije, diskutovati o njima i konsultovati se sa drugim ljudima.

Alati za modelovanje predstavljaju moćno sredstvo za analizu, dokumentovanje i razumavanje složenih poslovnih procesa. Pomoću njih možemo da:

- eliminišemo redundantne i neefikasne aktivnosti,
- smanjimo troškove,
- povećamo fleksibilnost sistema,
- poboljšamo kvalitet proizvoda i usluga.

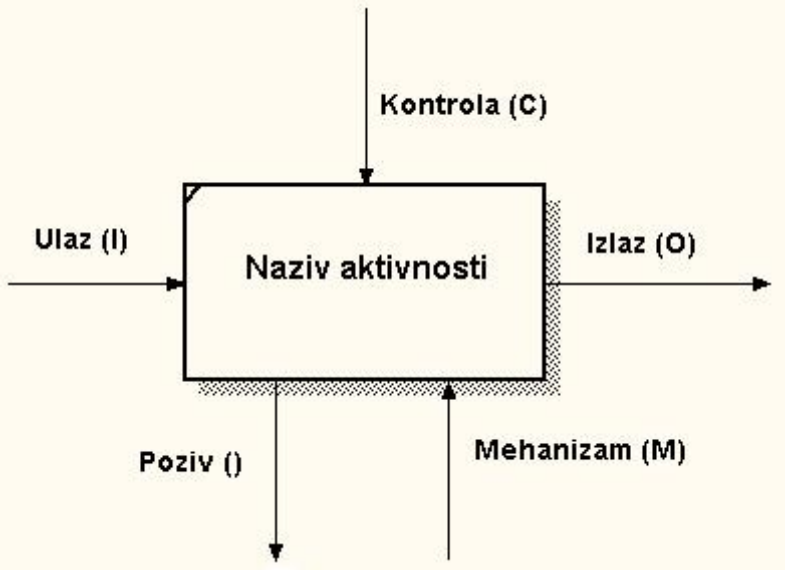
Korišćenjem ovih alata i dijagrama izrađenih u njima jasno je moguće predstaviti poslovne procese, njihove rezultate kao i resurse koji su potrebni da se oni izvrše. Dijagrami pružaju preciznu sliku o tome kako se pojedine aktivnosti sprovode, od najmanjih delova do cele organizacije.

U ovom radu se razmatra modelovanje procesa IDEF0 metodologijom. IDEF0 je tehnika koja se primenjuje u ranoj fazi razvoja projekta, a koristi se kao metodologija za analizu i logičko projektovanje. U toj tehnologiji sistem se predstavlja kao skup povezanih aktivnosti ili funkcija. Model je slikovito prikazan kao skup hijerarhijski ugnježdenih aktivnosti. Model aktivnosti posmatra sistem kao skup aktivnosti od kojih svaka transformiše neki objekat ili skup objekata. Aktivnosti (čine ih glagoli) se analiziraju nezavisno od objekata koji ih izvršavaju. Najviši nivo u hijerarhiji naziva se kontekstna aktivnost. Ona opisuje celokupni sistem. Nivoe ispod nje zovemo dekompozicionim nivoima. Oni predstavljaju podprocese akivnosti roditelja.

Prvi korak u izradi dijagrama aktivnosti je uočavanje najvišeg nivoa - kontekstne aktivnosti. Ona opisuje celokupni sistem. Obeležavamo je pravougaonikom kome dodeljujemo ime – glagol, a dodatno je opisujemo imenicom i/ili pridevom koja ga bliže objašnjava. Kontekstnu aktivnost razlažemo na skup dekompozicionih nivoa u kojima su aktivnosti predstavljene pravougaonicima, a za njihovo imenovanje važe ista pravila.

Dekompozicija se koristi u procesu modelovanja da bi se posmatrani sistem razložio na aktivnosti od kojih se sastoji. Neke od tako kreiranih (dekomponovanih) aktivnosti mogu dalje da se razlažu na činioce. Broj dekompozicionih nivoa zavisi od složenosti, kao i od potrebne detaljnosti razmatranja sistema.

Pravougaonici su povezani strelicama koje reprezentuju ljude, mesta, koncepte ili događaje. U IDEF0 dijagramima postoji četiri glavne vrste strelica: ulaz (engleski, Input), izlaz (Output), kontrola (Control) i mehanizam (Mechanism).



Slika 10: Pozicije strelica i njihova značenja u IDEF0 terminologiji

Ulazne strelice reprezentuju materijal ili informacije koje aktivnost upotrebljava ili transformiše da bi stvorila izlaze. One, u IDEF0 dijagramima, uvek ulaze u kvadrate aktivnosti sa leve strane, a može ih biti proizvoljan broj. Ulazi su opcioni jer neke aktivnosti mogu da ne vrše transformacije.

Izlazne strelice predstavljaju materijal ili informacije koje su proizvodi aktivnosti, pri čemu mora postojati bar jedan izlaz iz svake. Aktinost koja ne proizvodi izlaz ne bi trebala da se modeluje ili predstavlja kandidata za eliminaciju.

Strelice mehanizama prikazuju resurse koji sprovode aktivnosti. To mogu biti ljudi, oprema i alati koji obezbeđuju ili usmeravaju energiju da bi se aktivnost izvršila. Ovaj tip strelica može da se izostavi iz modela za određenu aktivnost ukoliko se ustanovi da nije neophodan.

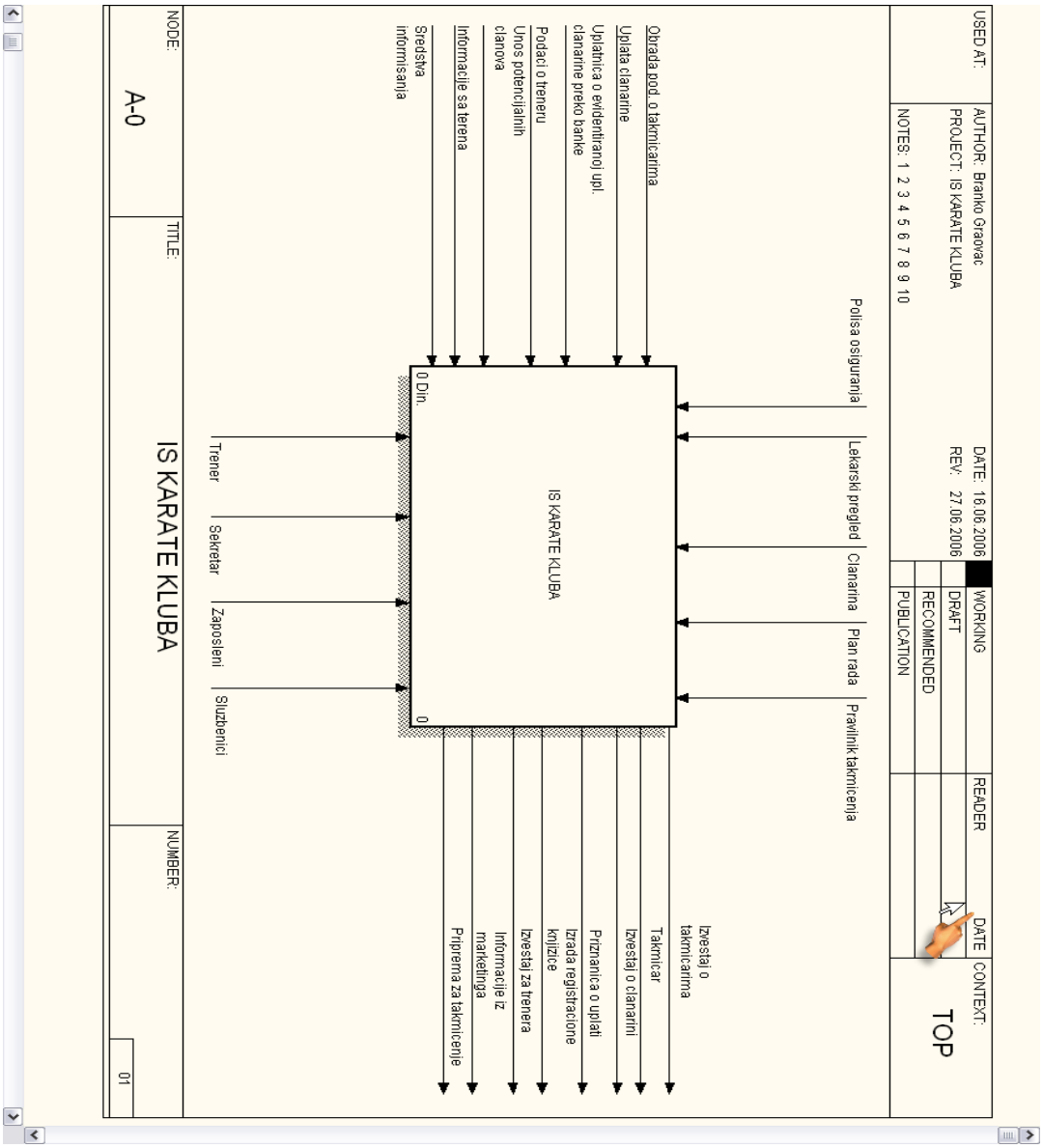
Strelice kontrola označavaju kontrole koje upravljaju ili regulišu način i vreme izvršavanja, kao i koje aktivnosti će se izvršiti da bi se dobili željeni (potrebni) izlazi. Svaka aktivnost bi trebala da ima bar jednu kontrolu. Obično su u obliku pravila, propisa, uredbi, zakona, normi ili standarda. Imaju uticaja na ulaze, bez da se transformišu ili troše. Često se koriste kao okidači koji potiču aktivnosti da počnu da se izvršavaju. One u dijagramima ulaze u aktivnosti sa gornje strane.

14 MODEL PROCESA – KARATE KLUBA

Program za karate klub je realizovan tako da obuhvata kompletnu evidenciju svih članova karate kluba, kako takmičara tako i trenera i ostalih zaposlenih. Posebno je obrađena evidencija takmičara kluba, od osnovnih podataka, položenih pojaseva, do postignutih rezultata na takmičenju. Iz programa se mogu odštampati određeni izveštaji kako za pojedinca, tako i za klub.

14.1 KONTEKSTNI DIAGRAM «IS karate kluba»

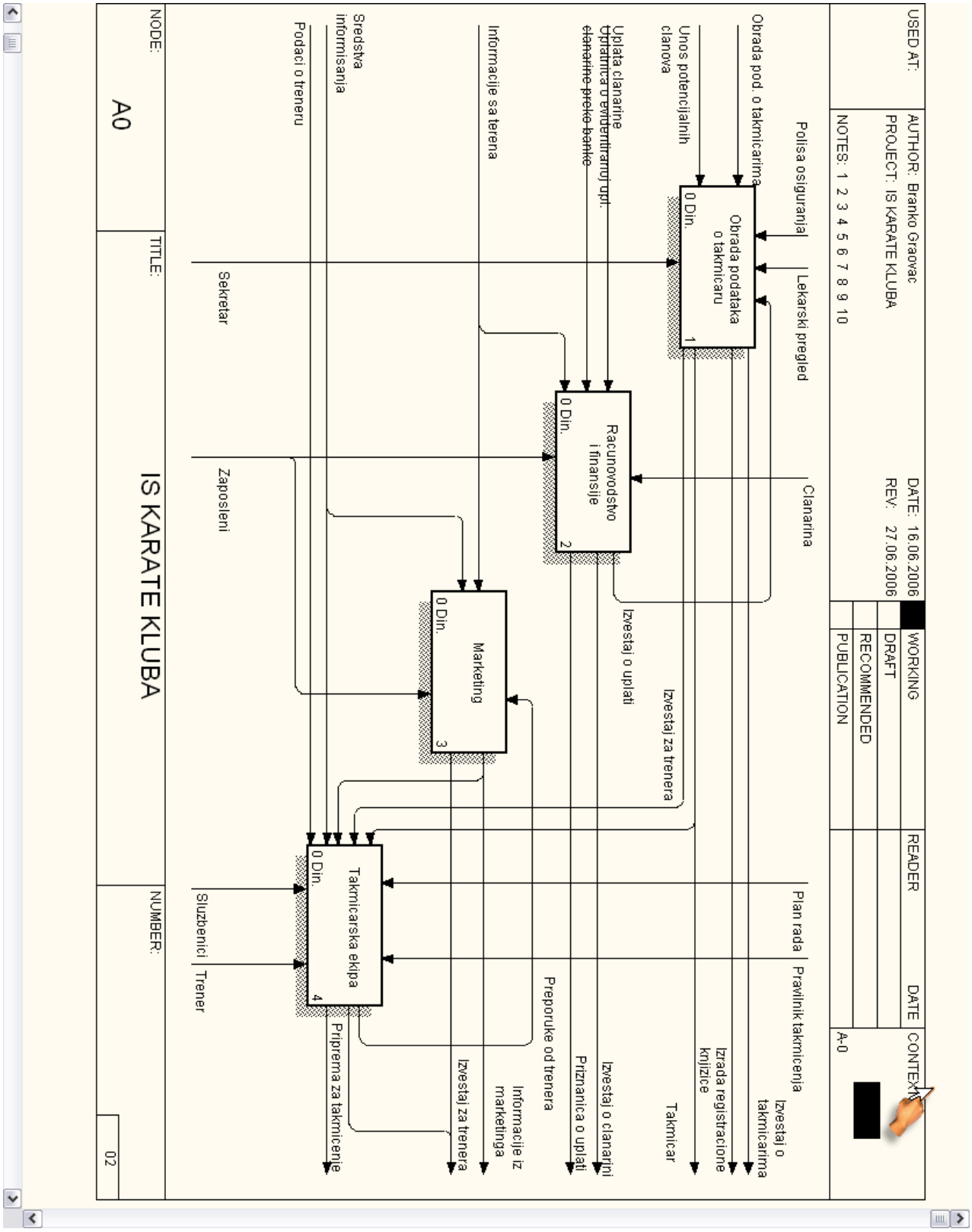
Koristeći prethodno navedenu sintaksu za izradu kontekstnog dijagrama, uz pomoć alata za modelovanje BPWin, kreiran je kontekstni dijagram za aktivnost "Karate kluba".



Slika 11 : Kontekstni dijagram za aktivnost "karate kluba"

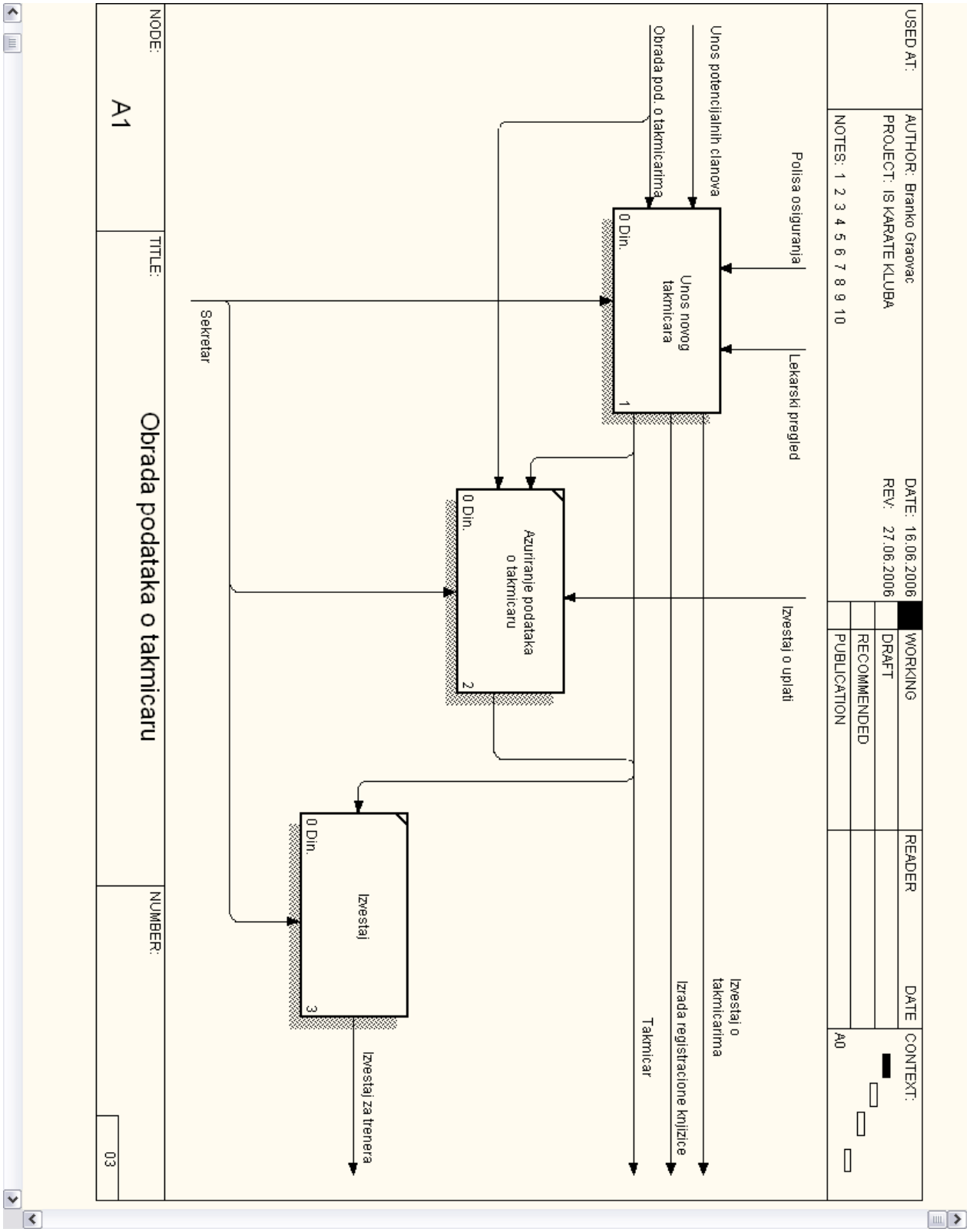
14.2 NULTI NIVO DEKOMPOZICIJE

Na osnovu kontekstnog dijagrama prikazanog na Slici 2, pristupa se dekompoziciji.



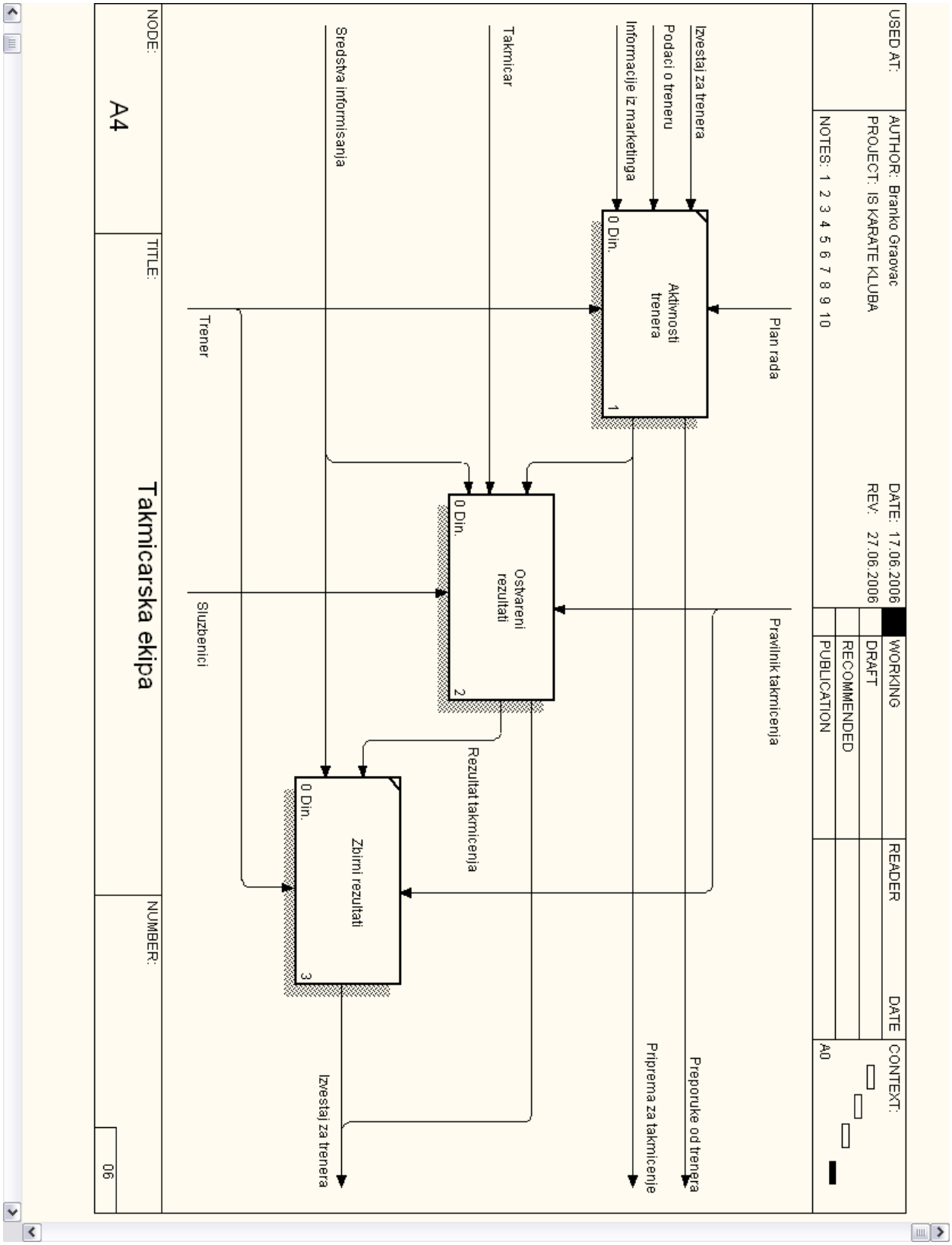
Slika 12 : Dekompozicioni dijagram za aktivnost "Karate kluba"

14.3 PRVI NIVO DEKOMPOZICIJE «obrada podataka o takmičaru»



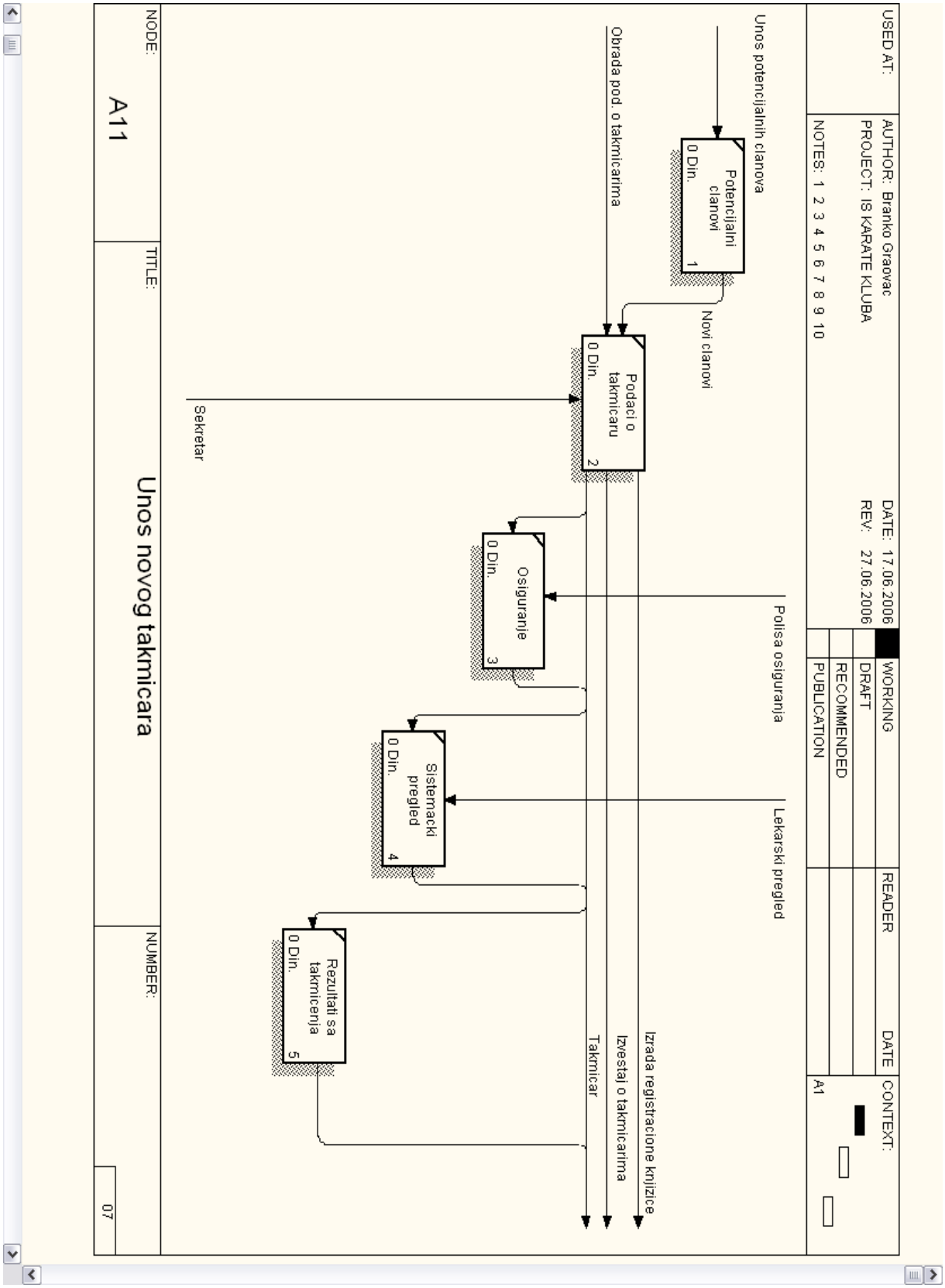
Slika 13 : Dekompozicioni dijagram za aktivnost "Obrada popdataka o takmičaru"

14.4 PRVI NIVO DEKOMPOZICIJE «takmičarska ekipa»



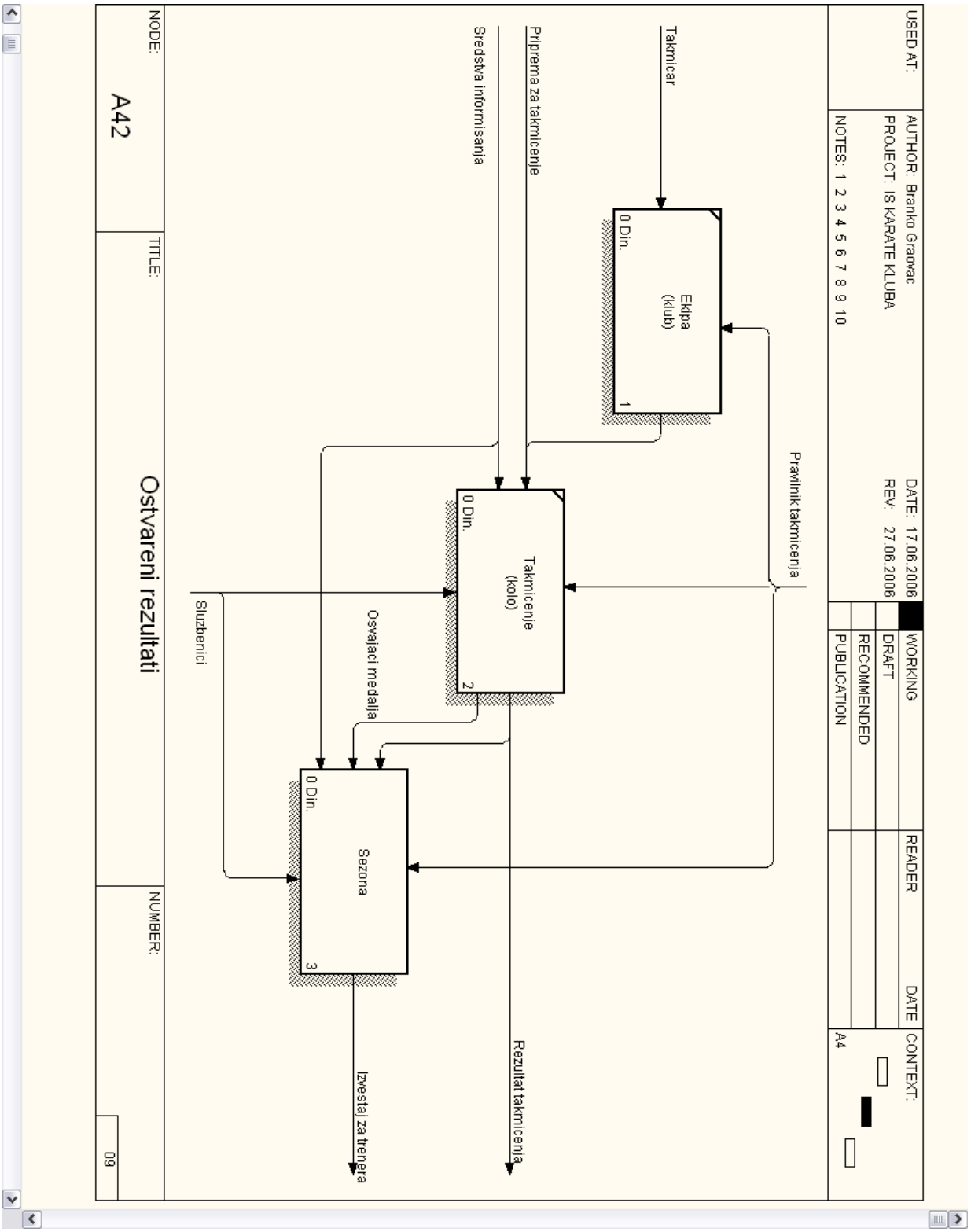
Slika 14 : Dekompozicioni dijagram za aktivnost "Takmičarska ekipa"

14.5 DRUGI NIVO DEKOMPOZICIJE «unos novog takmičara»



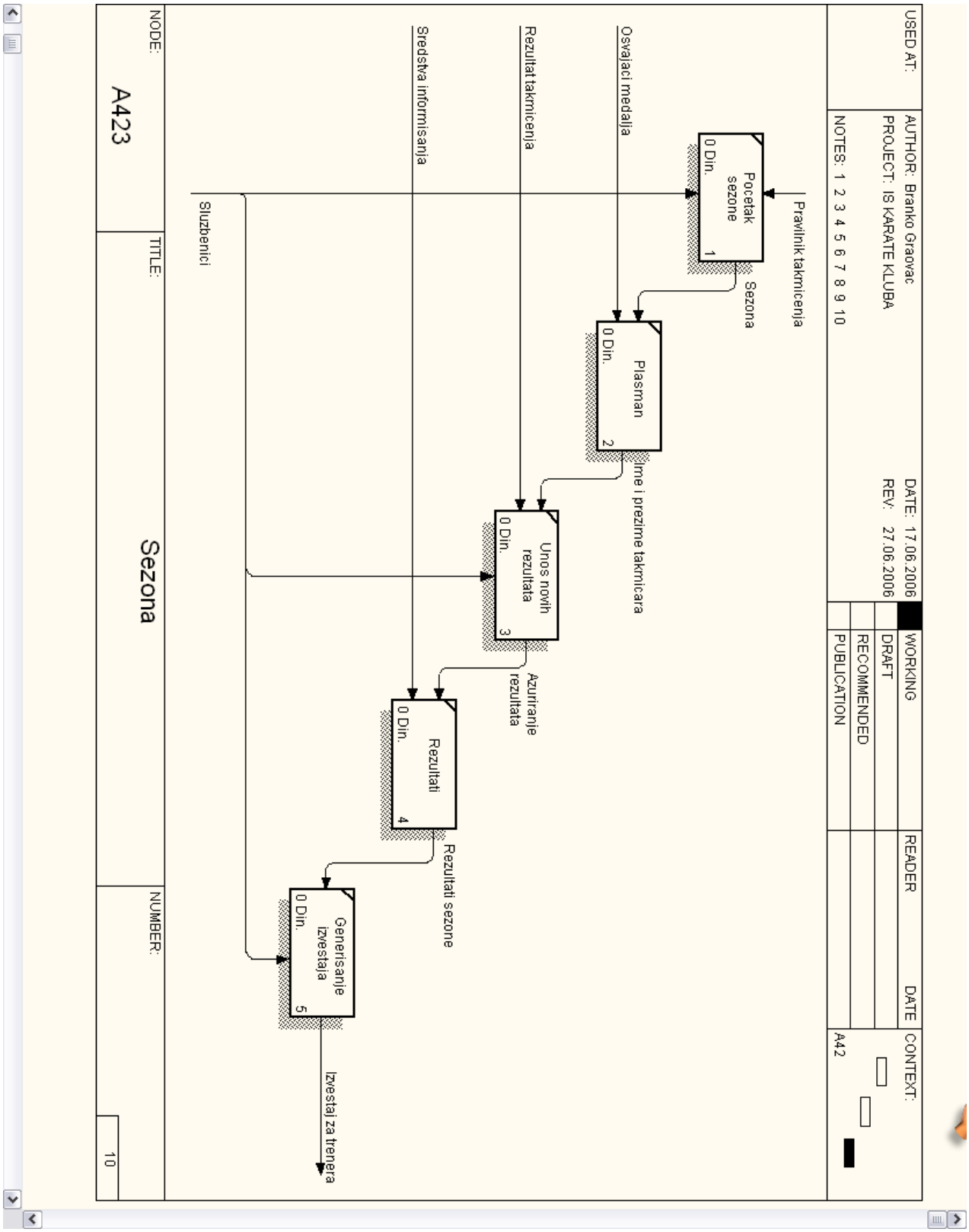
Slika 15 : Dekompozicioni dijagram za aktivnost "Unos novog takmičara"

14.6 TREĆI NIVO DEKOMPOZICIJE «ostvareni rezultati»



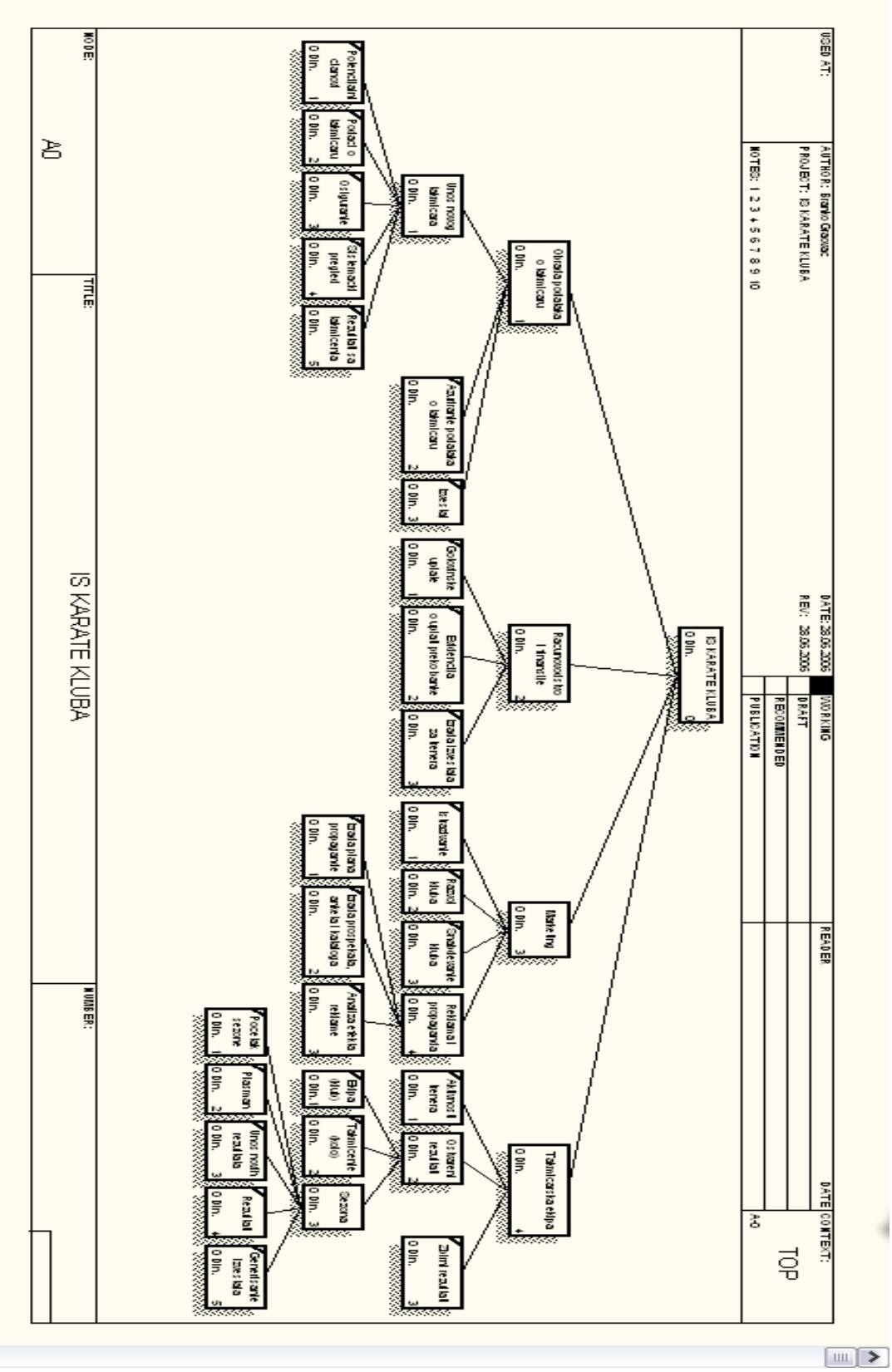
Slika 16 : Dekompozicioni dijagram za aktivnost "Ostvareni rezultati"

14.7 ČETVRTI NIVO DEKOMPOZICIJE «sezona»



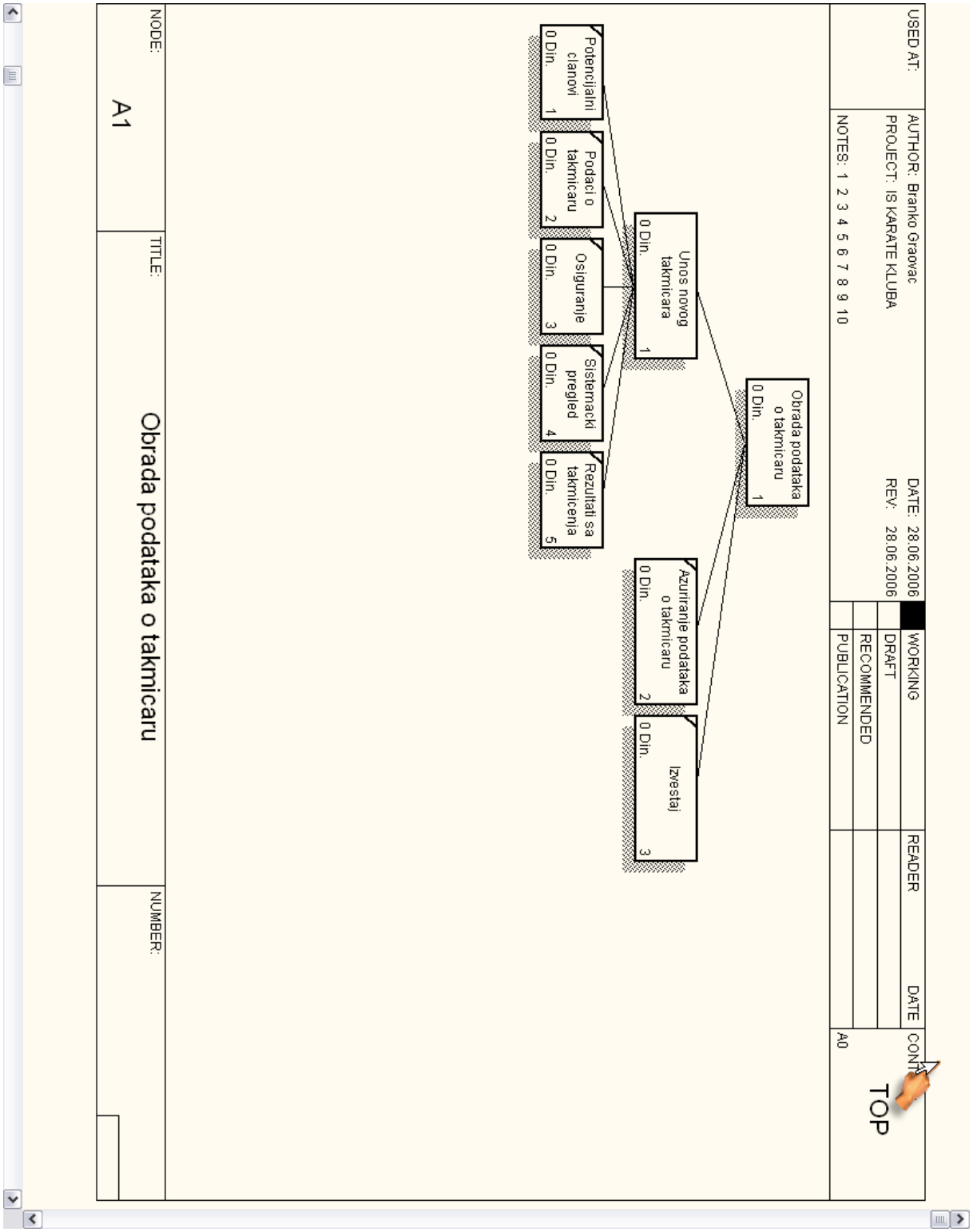
Slika 17 : Dekompozicioni dijagram za aktivnost "Sezona"

14.8 HIJERARHIJSKO STABLO PROCESA «IS karate kluba»



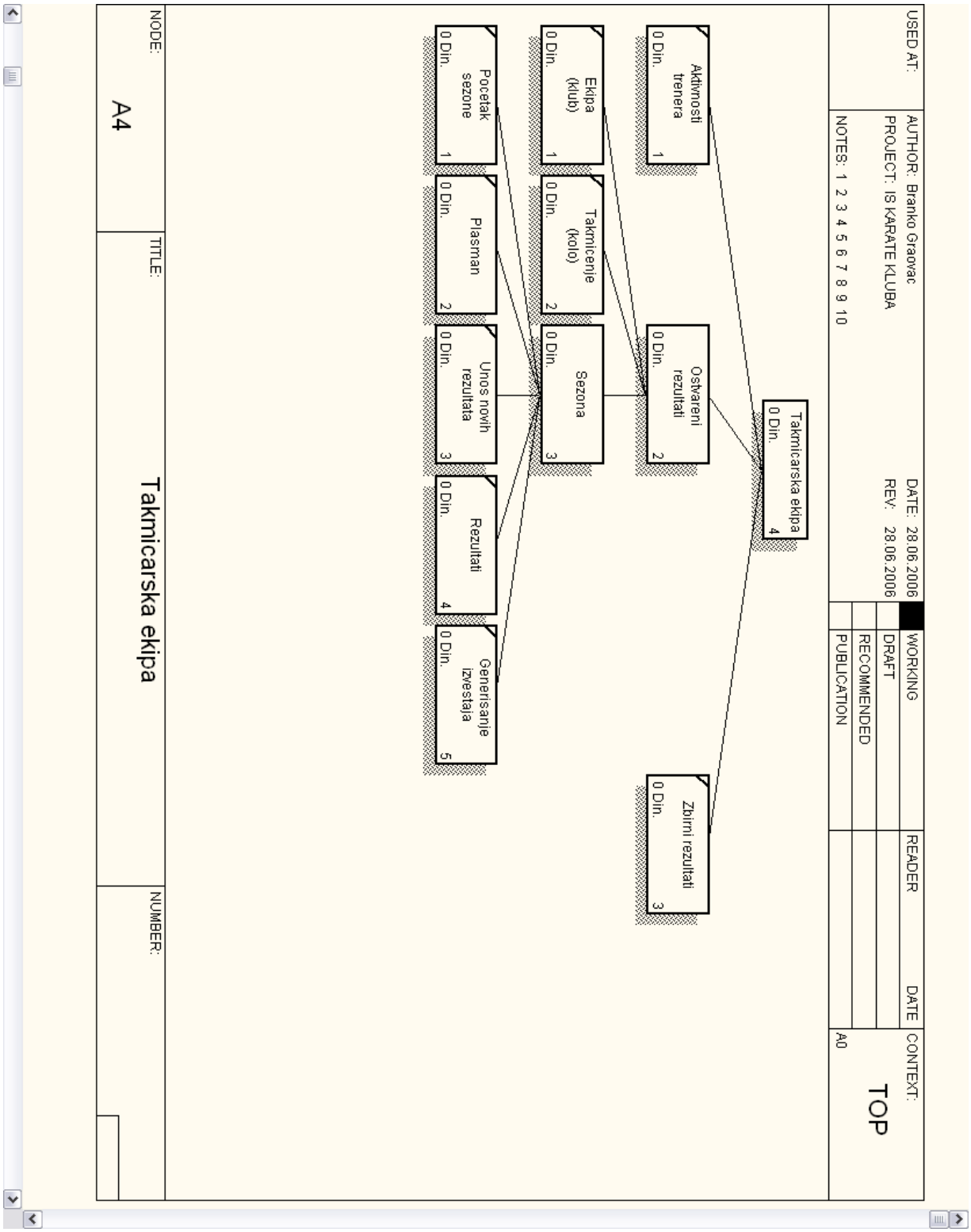
Slika 18 : Hijerarhijsko stablo procesa «IS karate kluba»

14.9 HIJERARHIJSKO STABLO PROCESA «obrada podataka o takmičaru»



Slika 19 : Hijerarhijsko stablo procesa «Obrada podataka o takmičaru»

14.10 HIJERARHIJSKO STABLO PROCESA «takmičarska ekipa»



Slika 20 : Hijerarhijsko stablo procesa «Takmičarska ekipa»



15 MODELOVANJE PODATAKA

Modelovanje podataka je proces u kome se definiše struktura podataka, odnosno pojednostavljeno predstavljanje posmatranog sistema preko skupa objekata ili entiteta, atributa objekata i veza između objekata. Model podataka reprezentuje stanje sistema u jednom trenutku vremena preko skupa podataka i njihovih međusobnih veza. Sadrži skup informacija o prošlosti i sadašnjosti sistema koje su potrebne da se pod dejstvom budućih poznatih ulaza mogu odrediti njegovi budući izlazi. Model se smatra potpuno određenim ako su definisane struktura podataka, kojima se definišu statičke karakteristike sistema, strukturna i vrednosna ograničenja i skup operacija koje definišu dinamičku interpretaciju podataka kroz njihovu obradu. Logički model zadovoljava potrebe projektanta koji modeluje poslovne informacije i poslovna pravila, a fizički administratora baze podataka koji je fokusiran na fizičku implementaciju projektovanog modela u konkretnoj bazi podataka. U radu je dat model podataka uz korišćenje IDEF1X notacije.

Entitet je osoba, stvar ili prostor o kojoj organizacija želi da ima podatke. Svaki entitet korespondira sa tabelom u bazi podataka. Objekti mogu biti zavisni ili nezavisni. Nezavisni objekti su oni čije instance mogu jedinstveno da se identifikuju bez njihovog povezivanja sa nekim drugim entitetom, tj. oni koji su autonomni od nekog drugog entiteta. U modelu podataka se prikazuju pravougaonikom u koji se upisuje njihov naziv. Zavisni su oni entiteti čija egzistencija i identifikacija zavise od drugih objekata. U modelu se interpretiraju kao pravougaonici sa zaobljenim ivicama i upisanim imenom.

Atribute odgovarajućeg objekta čine delovi informacije koje želimo da imamo o njemu, što u bazi podataka odgovara koloni u pripadajućoj tabeli. Mogu se nalaziti u oblasti ključeva ili u oblasti podataka.

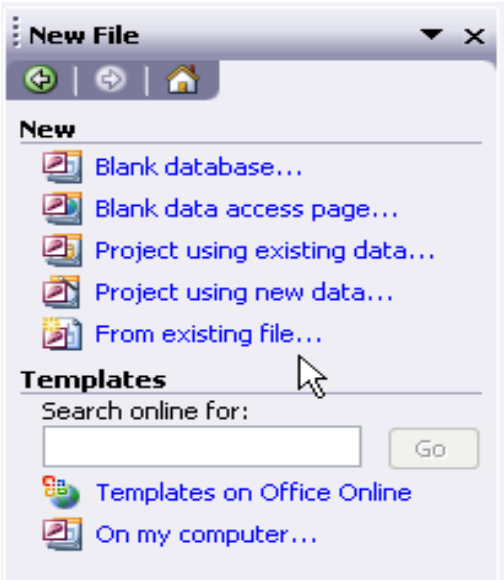
Veze delimo na identifikujuće, odnosno one koje entitet dete identifikuju kroz njegovu vezu sa entitetom roditelj, neidentifikujuće, kod kojih to nije slučaj, neodređene, kojima se smatraju veze više prema više i veze prema podtipovima koje uspostavljaju vezu između entiteta i njima zavisnih klasnih entiteta.

Svaki entitet mora da ima atribut ili grupu atributa koji jednoznačno identifikuju svaki njegov primerak, odnosno svako njegovo pojavljivanje. Taj atribut ili grupa nazivaju se primarni ključ (primary key). Ukoliko je u pitanju samo jedan atribut onda se on naziva prostim ključem, u protivnom, ključ je složen. Logično se zaključuje da ni jedan deo ključa ne može biti prazan, tj. njegova vrednost ne može biti NULL.

Pri kreiranju identifikujuće ili neidentifikujuće veze, atributi primarnog ključa entiteta roditelja migriraju u atribute entiteta dete kao spoljni ključ (foreign key). Kada se nekom entitetu doda spoljni ključ, stvara se zavisnost između objekta za koji se definiše spoljni ključ i objekta na koji se on odnosi. Time se stvara veza kojom, kada se unosi skup vrednosti atributa u entitet za koji se definiše spoljni ključ, mora postojati odgovarajući atribut u objektu na koji se spoljni ključ odnosi. Vrednost spoljnog ključa može biti i prazna, odnosno može biti NULL.

16 POKRETANJE PROGRAMA, PRIPREMNE RADNJE, KREIRANJE PRVE TABELE

Microsoft Access se pokreće kao i svaki drugi program u Windowsu - klikom (odnosno dvoklikom) na ikonu programa.



Slika 21: Blank database

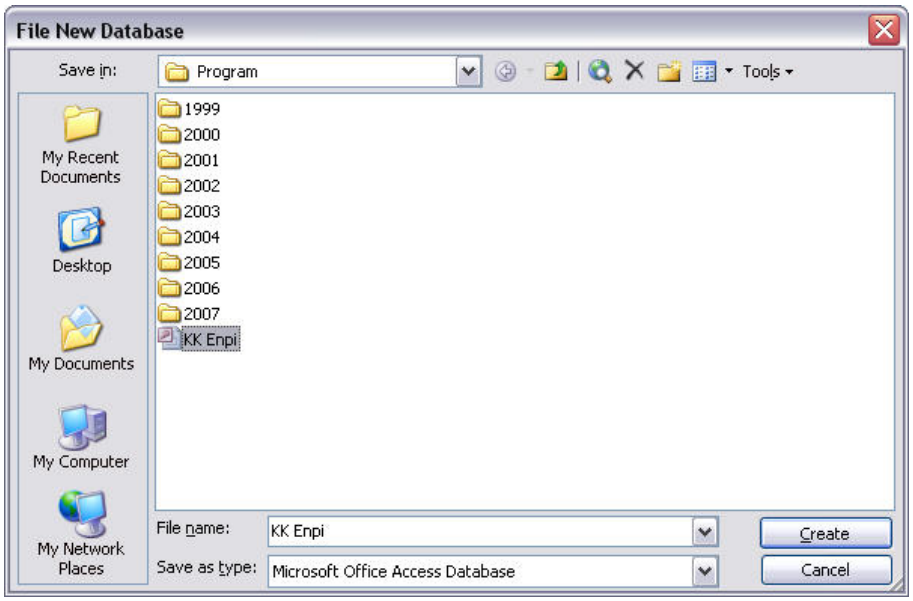
Posle pozdravnog ekrana aplikacije, pojavljuje se ekran na kome možete birati jednu od opcija: kreirati novu bazu (**Blank database**), iskoristiti jedan od čarobnjaka za automatizovano kreiranje baze biranjem jednog od predefinisanih tipova baza (**Access database wizards, pages, and projects**) ili otvoriti već postojeću bazu (**Open an existing file**).

Pošto kreiramo novu aplikaciju, biramo prvu opciju **Blank database**.

Posle klika na **Blank database**, sledi prvo iznenađenje - program traži da snimimo datoteku, iako još ništa nismo radili! Kod programa za rad sa bazama podataka integritet podataka je na prvom mestu. To znači da svaki podatak koji unesete, izmenite ili obrišete trenutno zapisuje na disk, a naredba **Save** pri tom nema isto značenje kao u drugim programima. Aktiviranjem naredbe **Save** snima se oblik prozora u kome radite, a ne sami podaci! Otuda zahtev za snimanjem datoteke ne predstavlja iznenađenje, već razumnu pojavu.

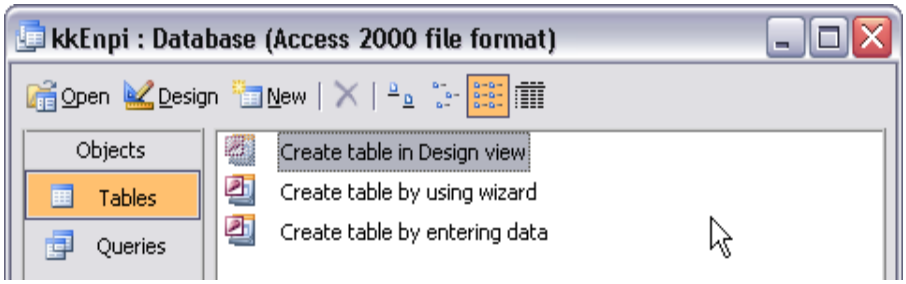


Ime datoteke će biti **KK Enpi**, a direktorijum u kojem je snimamo neka bude proizvoljan - a najbolje namenski, kreiran upravo za našu bazu.



Slika 22: Kreiranje kontejnera nove baze

Posle kreiranja datoteke, pojavljuje se prozor baze (u statusnoj liniji prozora će pisati **kkEnpi: Database**) sa komponentama (Tables, Queries, Forms, Reports, Macros i Modules).



Slika 23: Kreiranje nove tabele

Kreiranje baze počinje od kreiranja tabela i stoga polazimo logičnim redosledom.

16.1 Planiranje osnove

Izgradnja kuće počiva od temelja. Izgradnje baze počinje od osmišljavanja tabela i njihovog povezivanja. Za kreiranje programa **KK Enpi** za početak ćemo osmotriti celine - podatke koje treba grupisati u zasebnim tabelama. Svaki klub ima svoje članove - takmičare. Prva tabela će se stoga zvati **Takmicari**. Druga tabela će se zvati **Pojas**. Treća tabela će se zvati **Takmicenja**. Četvrta tabela **Statistika**, **itd**.

Zašto zasebne tabele? Jednostavno - zbog izbegavanja dupliranja podataka. Podaci kao što su telefon, adresa, ime i prezime su jedinstveni i ne postoji niti jedan razlog da se oni ponavljaju. Ako bi pak kreirali samo jednu tabelu, u nju bi morali da unosimo ime i prezime člana (njegovu adresu, broj telefona,...) i datum polaganja, takmičenja (sa svim podacima...), te bi ovim nepotrebno duplirali podatke. Još gore: veoma su moguće greške u kucanju - nikad ne bismo bili načisto na koliko je takmičenja bio neki takmičar, odnosno koliko je medalja osvojio.



16.2 Planiranje tabela

Pošto smo stvorili sliku o potrebnom broju tabela i njihovoj nameni, možemo preći na sledeći korak - definisanje tabela. Za svaku tabelu valja definisati polja i adekvatne tipove podataka koji se mogu u njih unositi:

Tabela 'Takmicari'

Tabela 2: Definicija tabele Takmičari

Naziv polja	Tip polja	Veličina polja	Komentar
ID_takmicar	Number	Integer (*)	broj karate pasoša
Prezime	Text	20	upisujemo prezime
Ime	Text	20	upisujemo ime
ID_pol	Text	1	Biramo ponuđenu vrednost M ili Ž
Slika	Text		
Adresa	Text	30	broj karaktera je restriktivno postavljen, ovo se kasnije može povećati po potrebi; uključuje naziv ulice i broj kuće
Mesto	Text	30	uključuje poštanski broj i naziv mesta
Telefon	Text	15	tip je tekst zbog interpunkcije formata xxx / AAA-BBBB
JBMG	Text	13	
God_rodj	Text	4	potrebno za statističku obradu starosne strukture članova
Dat_upisa	Date	Short Date(**)	takođe potrebno za statističku obradu
Napomena	Memo	(***)	

- (*) Integer je tip polja Number koji omogućava upis brojeva od -32.767 do 32.767 a kako identifikacioni broj ne može ići u minus smatramo tip polja 0 do 32.767; ako bi bilo više članova od tog broja, stavili bi tip Long Integer za brojeve od 0 do 2.147.483.648
- (**) Short Date predstavlja prikaz datuma u obliku dd/mm/yy
- (***) Memo je tip polja koji ne uzima fiksnu memoriju već zavisi od broja ukucanih karaktera

Tabela 'Zvanicna takmicenja'

Tabela 3: Definicija tabele Zvanična takmičenja

Naziv polja	Tip polja	Veličina polja	Komentar
ID_zvanicna_takmicenja	Number	Long Integer	
Zvanicana_takmicenja	Text	50	



Tabela 'Statistika takmicenja'

Tabela 4: Definicija tabele Statistika takmičenja

Naziv polja	Tip polja	Veličina polja	Komentar
ID_statistika	AutoNumber	Long Integer	
ID_takmicar	Number	Long Integer	
Datum_takmicenja	Date/Time		
KP_plasman	Text	1	Upisujemo 1, 2 ili 3 mesto
BP_plasman	Text	1	Upisujemo 1, 2 ili 3 mesto

Tabela 'Statistika pojas'

Tabela 5: Definicija tabele Statistika pojas

Naziv polja	Tip polja	Veličina polja	Komentar
ID_pojas_statistika	AutoNumber	Long Integer	
ID_takmicar	Number	Long Integer	
ID_pojas	Number	Long Integer	
Datum_polaganja	Date/Time		
ID_ocena	Text	1	
ID_predsednik_komisije	Number	Long Integer	

Tabela 'Pojas'

Tabela 6: Definicija tabele Pojas

Naziv polja	Tip polja	Veličina polja	Komentar
ID_pojas	Number	Long Integer	
Pojas	Text	20	

Tabela 'Pol'

Tabela 7: Definicija tabele Pol

Naziv polja	Tip polja	Veličina polja	Komentar
ID_pol	Number	Long Integer	
Pol	Text	1	

Tabela 'Pol'

Tabela 8: Definicija tabele pol

Naziv polja	Tip polja	Veličina polja	Komentar
ID_mesto	Number	Long Integer	
Mesto	Text	20	
ID_Drzava	Number	Long Integer	



Tabela 'Polisa'

Tabela 9: Definicija tabele Polisa

Naziv polja	Tip polja	Veličina polja	Komentar
ID_polisa	Text	30	
Datum_izdavanja	Date/Time		
Ukupan_iznos	Currency		
ID_takmicar	Number	Long Integer	
ID_tip_osiguranja	Text	20	
Broj_mesecnih_rata		Integer	
Premijski_razred	Number	Integer	

Tabela 'Tip osiguranja'

Tabela 10: Definicija tabele Tip osiguranja

Naziv polja	Tip polja	Veličina polja	Komentar
ID_tip	Text	20	
Naziv	Text	20	
Tarifa	Number	Double	

Tabela 'Lekarski pregled'

Tabela 11: Definicija tabele Lekarski pregled

Naziv polja	Tip polja	Veličina polja	Komentar
ID_pregled	AutoNumber	Long Integer	
ID_takmicar	Number	Long Integer	
ID_doktor	Number	Long Integer	
ID_vrsta_pregleda	Text	20	
Datum_pregleda	Date/Time		
Napomena	Memo		

Tabela 'Doktor'

Tabela 12: Definicija tabele Doktor

Naziv polja	Tip polja	Veličina polja	Komentar
ID_doktor	Number	Long Integer	
Doktor	Text	30	

Tabela 'Vrsta pregleda'

Tabela 12: Definicija tabele Vrsta pregleda

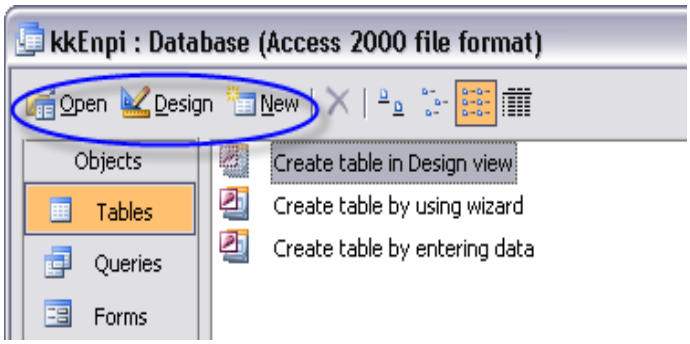
Naziv polja	Tip polja	Veličina polja	Komentar
ID_vrsta_pregleda	Number	Long Integer	
Pregled	Text	30	

16.3 Po istom principu će mo kreirati i ostale tabele.

Ovim smo definisali tabele i tipove podataka. Mesta za reorganizaciju ima još uvek, možemo kreirati zasebnu tabelu mesta i u nju upisati sva moguća mesta u kojima se održavaju takmičenja.

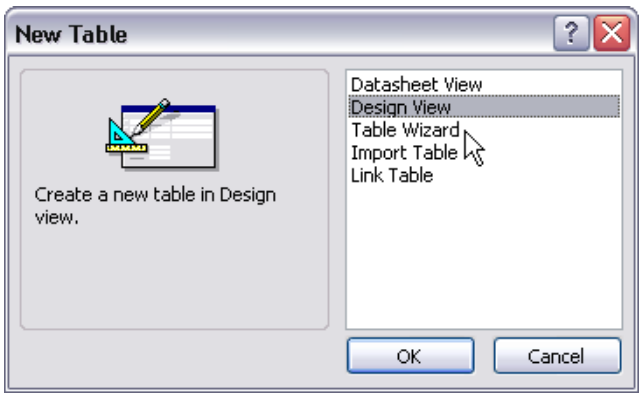
U prozoru baze strana **Tables** je već selektovana.

U vrhu ekrana prozora baze se nalaze tri tastera: **Open** (otvara postojeću tabelu za pregled), **Design** (izmene u strukturi tabele) i **New** (kreiranje nove tabele).



Slika 24: Mogućnosti manipulacije tabelama

Biramo taster **New**. Od opcija koje se pojavljuju u sledećem prozoru dijaloga koristićemo drugu - **Design View**.



Slika 25: Opcije za kreiranje nove tabele

Sve opcije u ovom prozoru imaju sledeća značenja:

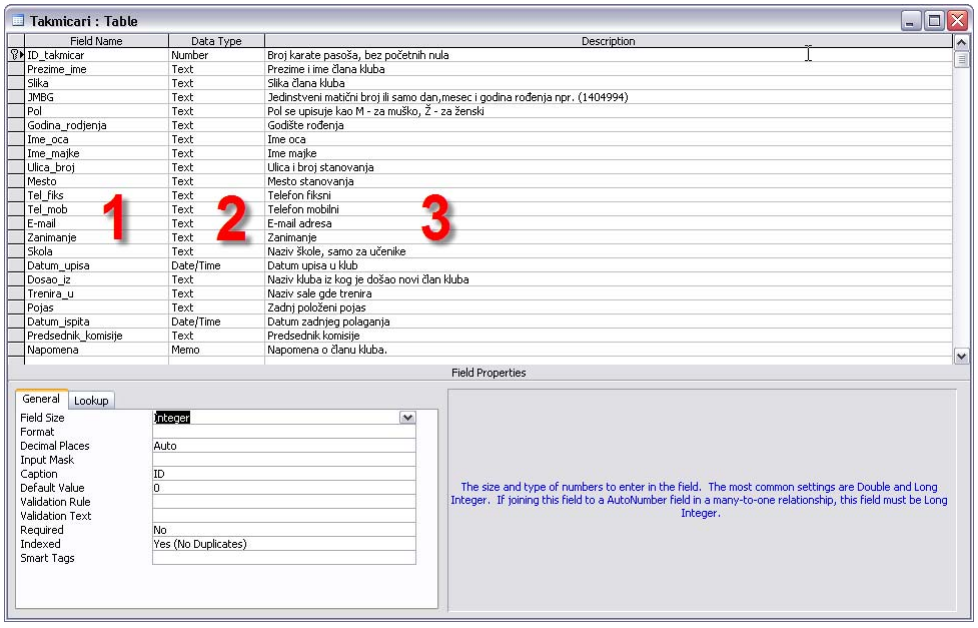
- **Datasheet View** otvara nedefinisanu tabelu sa univerzalnim poljima u koje se mogu unositi podaci, a Access će pokušati sam da postavi tip polja na osnovu unetih podataka;
- **Design View** predstavlja pogled na tabelu u prozoru se dizajn. U ovom prozoru zadajemo nazive kolone i adekvatne tipove podataka koji se mogu nalaziti u kolonama, odnosno definišemo broj kolona i tipove podataka;
- **Table Wizard** je čarobnjak koji će na osnovu vaših odgovora na postavljena pitanja kreirati tabelu;
- **Import Table** služi za unos podataka iz već postojeće tabele ili baze. Ovim se podaci kopiraju iz ulazne tabele u novokreiranu tabelu;



- **Link Table** postavlja prečicu koja pokazuje na drugu tabelu iz otvorene baze sa kojom trenutno radimo ili druge Access baze. Ovim se podaci ne kopiraju niti premeštaju, već se uspostavlja relacija na povezanu (linkovanu) tabelu.

Pošto smo selektovali opciju **Design View** i potvrdili izbor klikom na taster OK, otvara se prozor za dizajniranje tabele.

Slika 26: Design prozor tabele



Na slici, kolona obeležena brojem 1 predstavlja naziv kolone, broj 2 je tip kolone, a broj 3 je komentar koji se pojavljuje u statusnoj liniji Accessovog prozora.

U **Field Name** preseka prve kolone i reda upisaćemo tekst **ID_takmicar**. U drugoj koloni istog reda ukucaćemo **Number**. U koloni broj 3 istog reda možemo staviti komentar, koji se pojavljuje u Accessovoj statusnoj liniji.

Primetimo da se na delu ekrana iznad kojeg piše **Field Properties** pojavila tabela sa različitim stavkama.

Još jednom da objasnimo šta smo učinili. Započeli smo kreiranje tabele i nazivu prve kolone smo dodelili tekst **ID_takmicar**. Tip polja, kako smo ranije isplanirali, je **Number**. To znači da će u polje moći da se ukucavaju isključivo numeričke vrednosti, a sam tip polja ćemo postaviti na **Integer**. Na slici, ispod teksta **Field Properties**, na kartici **General** prva opcija je "veličina" (odnosno pod-tip) polja.

Osobine koje se mogu postaviti za polje Number su odozgo na dole sledeće:

- **Field Size** - veličina, odnosno tip polja
- **Format** - način na koji će ukucani brojevi biti prikazani
- **Decimal Place** - broj decimala u cifri. Vrednost postavljamo na nula, pošto identifikacioni broj nema smisla sa decimalama
- **Input Mask** - "maska" za unos brojeva. Ako u ovo polje postavimo vrednost **00000**, korisnik će morati da unese tačno pet cifara za identifikacioni broj (npr. **00023**)



- **Caption** - tekst koji se pojavljuje kao naziv kolone u pogledu za unos podataka
- **Validation Rule** - pravilo koje se mora ispoštovati pri unosu. Vrednost smo postavili na **>0** kako se ne bi mogli uneti negativni brojevi.
- **Validation Text** - tekst koji se pojavljuje kao upozorenje kada korisnik pogreši pri unosu
- **Required** - određuje da li je unos neophodan ili ne. Vrednost postavljamo na **Yes**.
- **Indexed** - da li je polje indeksirano ili ne. U indeksiranom polju pretraga je brža. **No Duplicates** u zagradi znači da se ne smeju pojaviti (ukucati) dve iste šifre za dva različita takmičara.

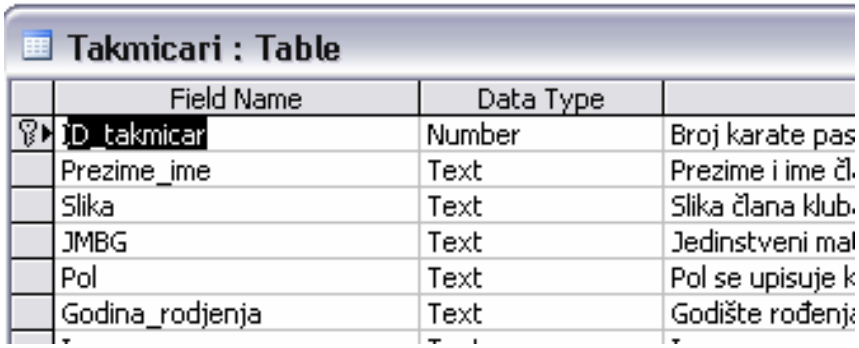
Sledeći i poslednji važan korak je postavljanje primarnog ključa. Primarni ključ je jedno ili više polja preko kojih će tabela biti povezana sa drugim tabelama, preko koje će se podaci indeksirati i izbeći mogućnost dupliranih unosa. U tabeli **Takmicari** primarni ključ ćemo postaviti na polje **ID_takmicar**. Pozicionirajmo kurzor na red u kome se nalazi tekst **ID_takmicar**. Kliknimo na ikonu ključa na toolbaru:

Slika 27: Primarni ključ



Izgled reda tabele **ID_takmicar** će biti sada ovakav:

Slika 28: Polje primarnog ključa



Ostaje nam još da snimimo tabelu. Iz menija **File** odaberimo opciju **Save**. U novootvorenom dijalogu upišimo tekst **Takmicari** (**bez slova Č u njemu!**).

Slika 29: Memorisanje tabele



Ovim je prva tabela kreirana i snimljena. Na sličan način ćemo kreirati i ostale tabele.



17 POVEZIVANJE TABELA

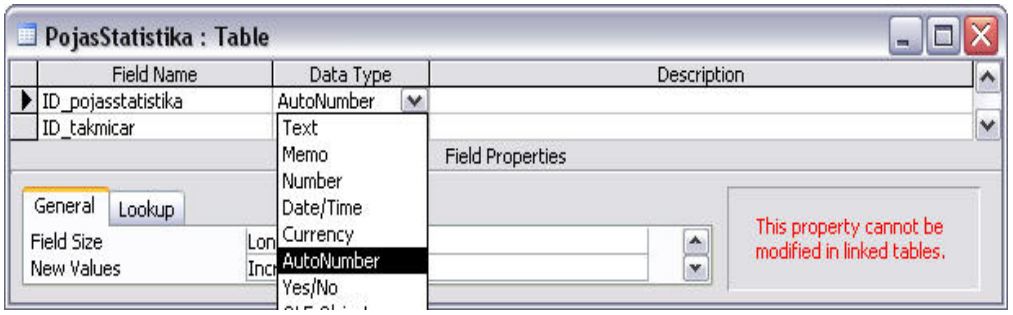
Iskoristićemo postojeće, kreirane tabele i povezaćemo ih međusobno u relacije. Punu snagu Access ostvaruje iz relacionih tabela (otuda naziv relacione baze podataka). Iz relacionih tabela podatke je lako filtrirati i na njih primeniti razne kriterijume bez obzira iz koliko tabela podaci potiču.

Ako analiziramo spisak polja i strukturu tabela videćemo da međusobnog povezivanje tabela nema mnogo. Polje **ID_takmicar** bi valjalo povezati sa istoimenim poljem iz tabele **Statistika i PojasStatistika,** polje **ID_takmicenja** sa istoimenim poljem iz tabele **Statistika**.

Verovatno ni u jednom programu nema više pomoćnih procedura koje Microsoft naziva čarobnjacima nego u Accessu. Pokrenimo proceduru ...

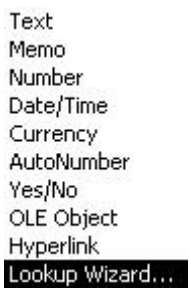
Selektujemo tabelu **PojasStatistika** i kliknemo na taster **Design**. Sada se otvara već poznati dizajn-pogled tabele. Kliknimo u polje **Number** reda **ID_takmicar**. Primetićemo da je polje "drop-down" tipa, odnosno da sadrži listu za izbor koja se može videti klikom na strelicu na dole ili pritiskom na taster F4.

Slika 30: polje "drop-down" tipa

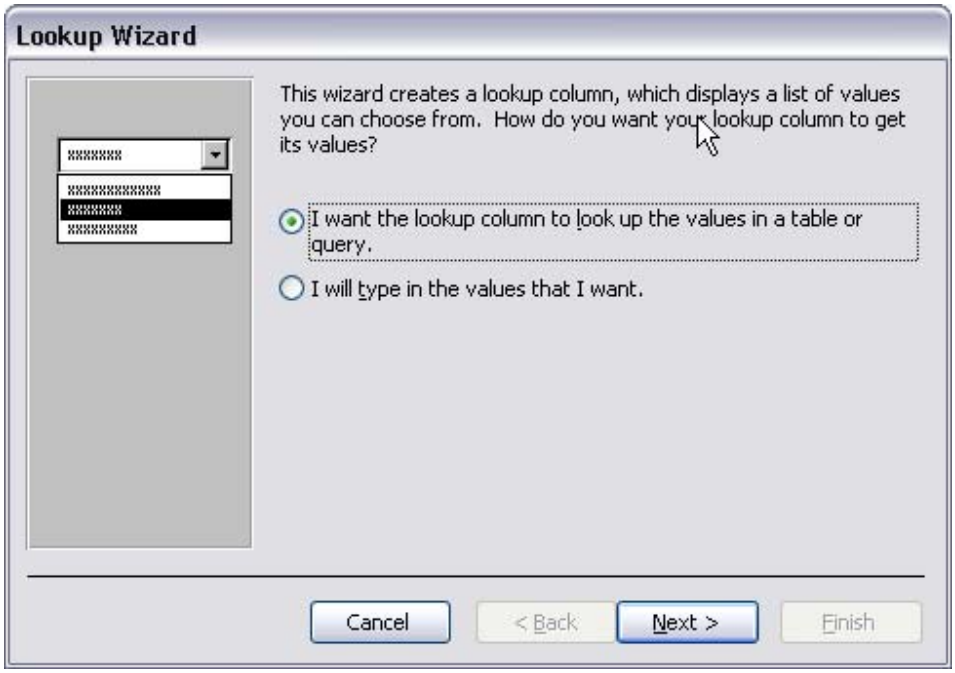


Ono što ćemo sada učiniti je da polja **ID_takmicar** ograničimo na sadržaj tabele **Takmicari**. Tip polja **Number** reda **ID_takmicar** promenimo na **Lookup Wizard**...

Slika 30a: Izbor opcije Lookup Wizard



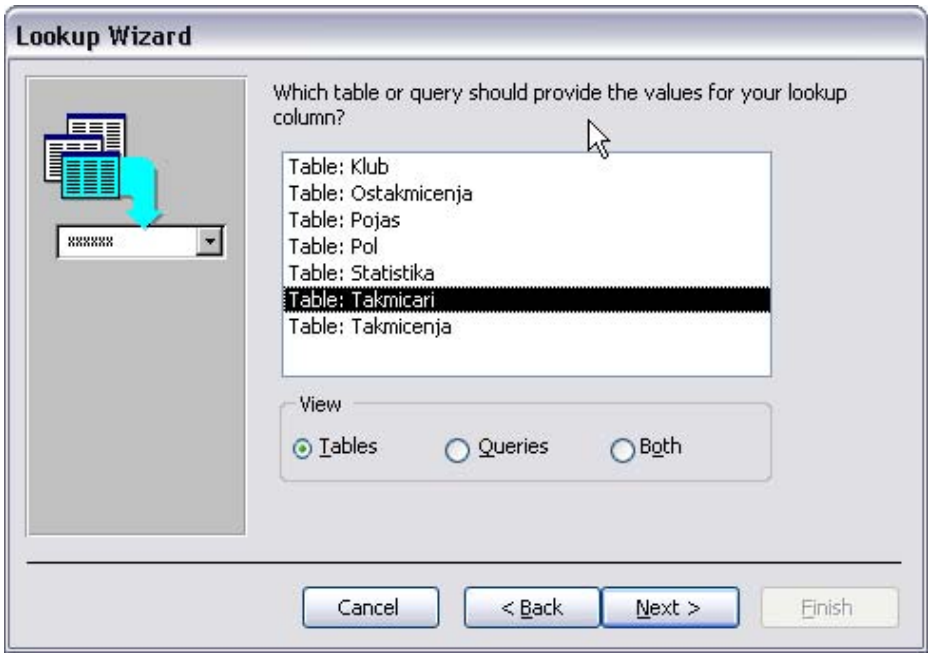
Pokreće se procedura koja će na osnovu naših odgovora na postavljena pitanja formirati relaciju umesto nas:



Slika 31: Početak procedure Lookup Wizard

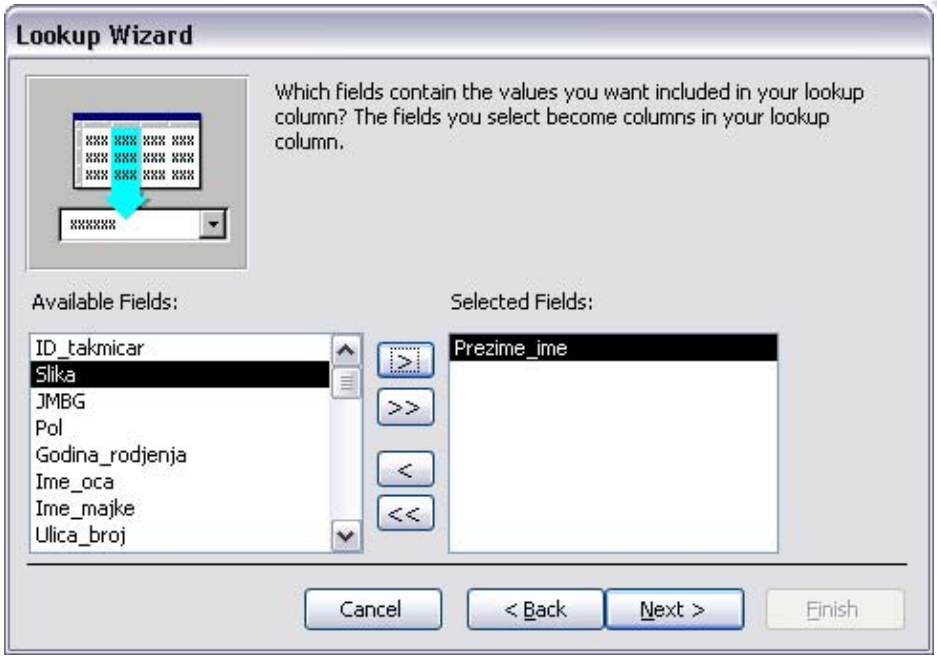
Na prvo pitanje odgovaramo sa prvom opcijom jer možemo da se u tabeli **PojasStatistika** na mesto prezime i ime pojavljuju

određene vrednosti iz tabele takmicari, odnosno samo identifikacioni brojevi. Klikom na **Next** (ili sa tastature **ALT+N**) sledi drugi korak:



Slika 32: Biramo odgovarajuću tabelu

U ovom koraku biramo tabelu iz koje će se uzimati podaci, odnosno određujemo "izvorišnu" tabelu.

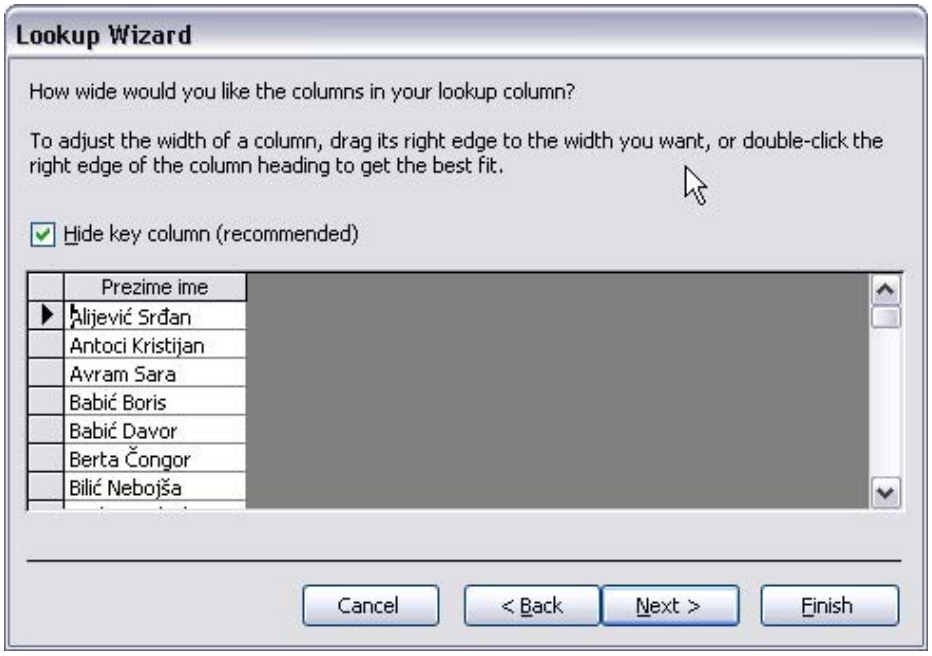


Slika 33: Biramo odgovarajuću kolonu

Dalje, u trećem koraku određujemo kolonu (polja) koja će biti "izvor" podataka:

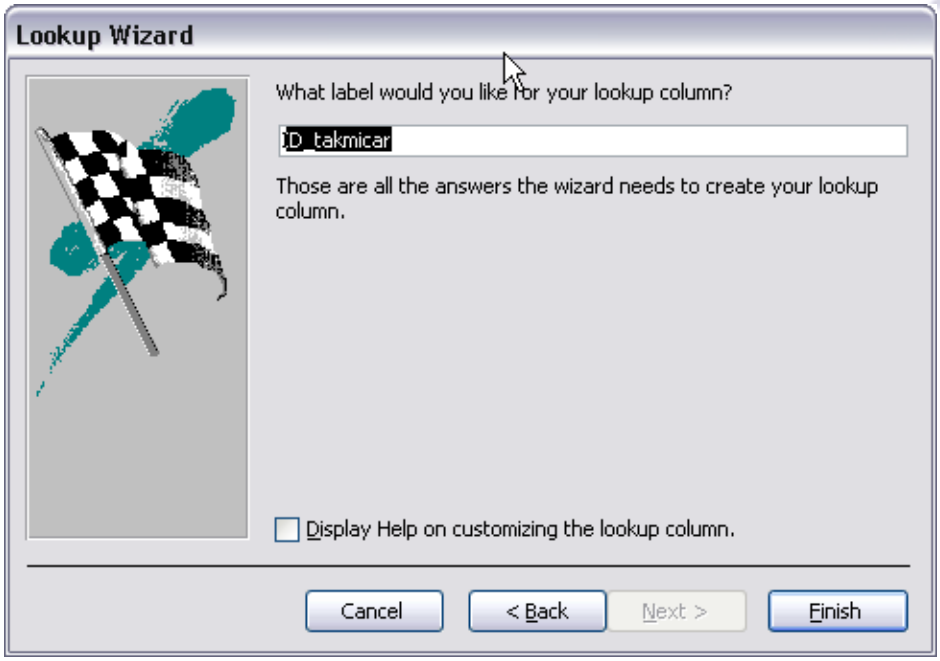
Preostala dva koraka nisu neophodna i već sada možemo kliknuti na taster **Finish**.

Četvrti korak je "kozmetički" i ovde određujemo širinu kolone koje će se prikazivati na ekranu:



Slika 34: Vidimo sadržaj izabrane kolone

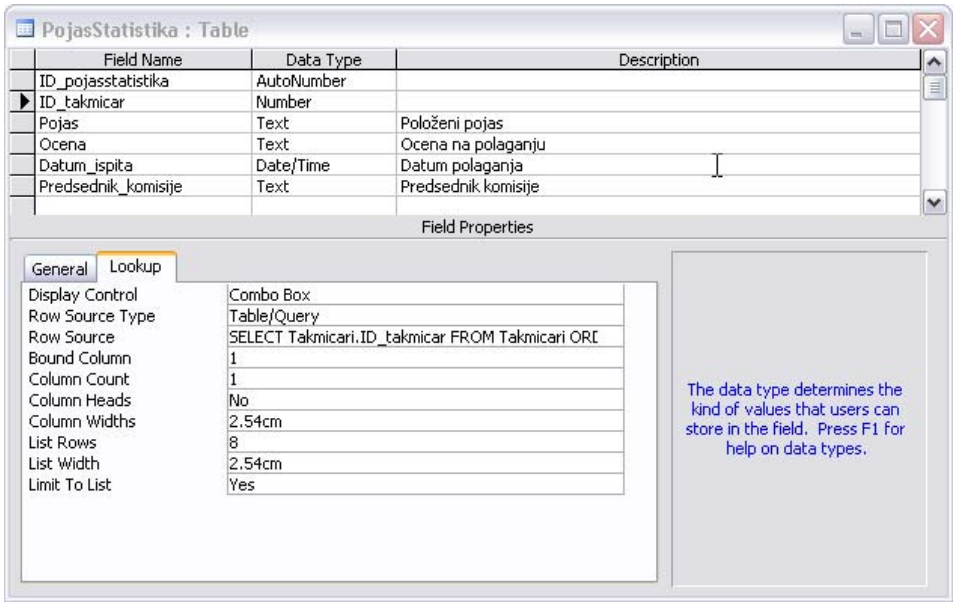
U poslednjem koraku možemo uticati na naziv kolone, a kako nam naziv **ID_takmicar** već odgovara, kliknućemo na taster **Finish**:



Slika 35: Kraj procedure Lookup Wizard

Ovim je kreirana prva tzv. relacija tabela **Takmicari** i **PojasStatistika**.

Još jednu, veoma bitnu, stvar moramo napraviti da bi ova veza funkcionisala kako valja: operateru se mora onemogućiti da u polja **ID_takmicar** i upisuje proizvoljne vrednosti, odnosno one vrednosti za koje ne postoji odgovarajuća šifra ili ime u tabeli **Takmicari**. Otvorimo ponovo tabelu **PojasStatistika** u dizajn-pogledu. Selektujmo red **ID_takmicar** klikom miša. Ispod teksta **Field Properties** postoje dva tzv. taba - **General** i **Lookup**. Klikom na tab **Lookup** otvara se lista osobina za polja čiji sadržaj se gleda iz druge tabele. Na ovoj listi za polje **ID_takmicar** osobinu **Limit To List** postavimo na vrednost **Yes**.



Slika 36: Lookup osobine

Snimimo i zatvorimo tabelu. Ovim smo ograničili sadržaj polja **ID_takmicar** na sadržaj druge relevantne tabele - **Takmicari**. Istu proceduru možemo ponoviti za polje **Pojas** i **Ocena**.

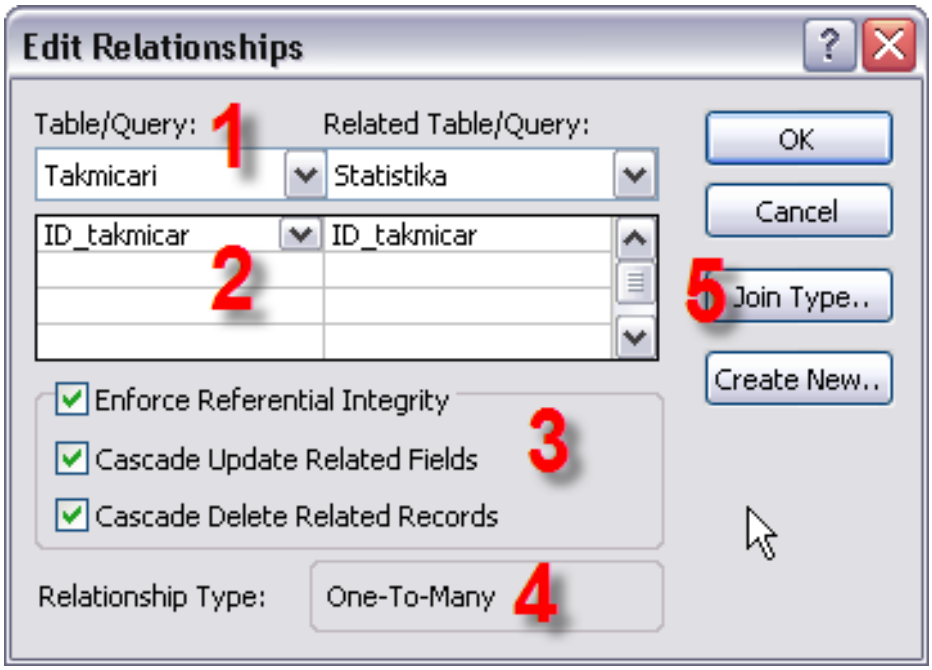
Pored ovih **Lookup** relacija napravićemo još i prave relacije između tabela. Iz menija **Tools** biramo opciju **Relationships**.... U prazan prozor koji se pojavi, dodaćemo tabele koje ćemo povezati. Spisak postojećih tabela dobija se klikom na taster **Show Table**.

Slika 37: Ikonica Relationships



Dodajmo redom tabele: **Takmicari**, **Pol**, **Pojas**, **PojasStatistika**, **Takmicenja**, **Ostakmicenja** i **Statistika**. Iz tabele **Takmicari** prevucimo polje **ID_takmicar** u tabelu **Statistika** na polje **ID_takmicar**. U prozoru koji se pojavi, podešavamo osobine ove relacije.

Slika 38: Prozor Edit Relationships



Na slici, obeleženo brojem 1 su tabele koje će biti povezane. Polja koja su međusobno povezana su obeležena brojem 2. Pod brojem 3 su opcije koje će definisati ponašenje ciljne tabele u zavisnosti od promene podataka u izvorišnoj tabeli. Ove opcije ćemo sve uključiti jer će one omogućiti da se promena stanja u tabeli **Takmicari**, reflektuje na tabelu **Statistika**.



Primer

Ako je takmičar sa sa određenim imenom i prezimenom učestovao na nekom takmičenju, ove promene će ostati zapisana u tabeli **Statistika** i taj podatak može poslužiti u vođenju statistike o učešću članova kluba na pojedinim takmičenjima. Takođe, ukoliko neki rakmičar osvoji medalju na nekom takmičenju, može se na osnovu istorije takmičenja uvek utvrditi gde je ko osvojio određenu medalju.

Na slici pod brojem 4 označen je tip relacija. U našem slučaju **One-to-many** znači da svaki takmičar može imati jedinstven broj, da se može više puta takmičiti, te da jedanom takmičenju može uzeti više medalja. Sem ove relacije postoji još dva tipa relacije. Druga je **One-to-one** što bi značilo da na jednom takmičenju miže učestovati samo jedan takmičar. Ova relacija bi bila primenljiva u, recimo, "Sega" klubovima, gde biste jednom članu dozvolili da iznajmi jednu konzolu za igranje. Treća vrsta relacija je **Many-to-many**. Ova relacija se upotrebljava u slučajevima kada postoji osobina zajedničke pripadnosti. Za primer, recimo da dva radnika zaduže dva kompleta alata bez posebne zajedničke pripadnosti. Obojica su tada vlasnici i jednog i drugog kompleta alata.

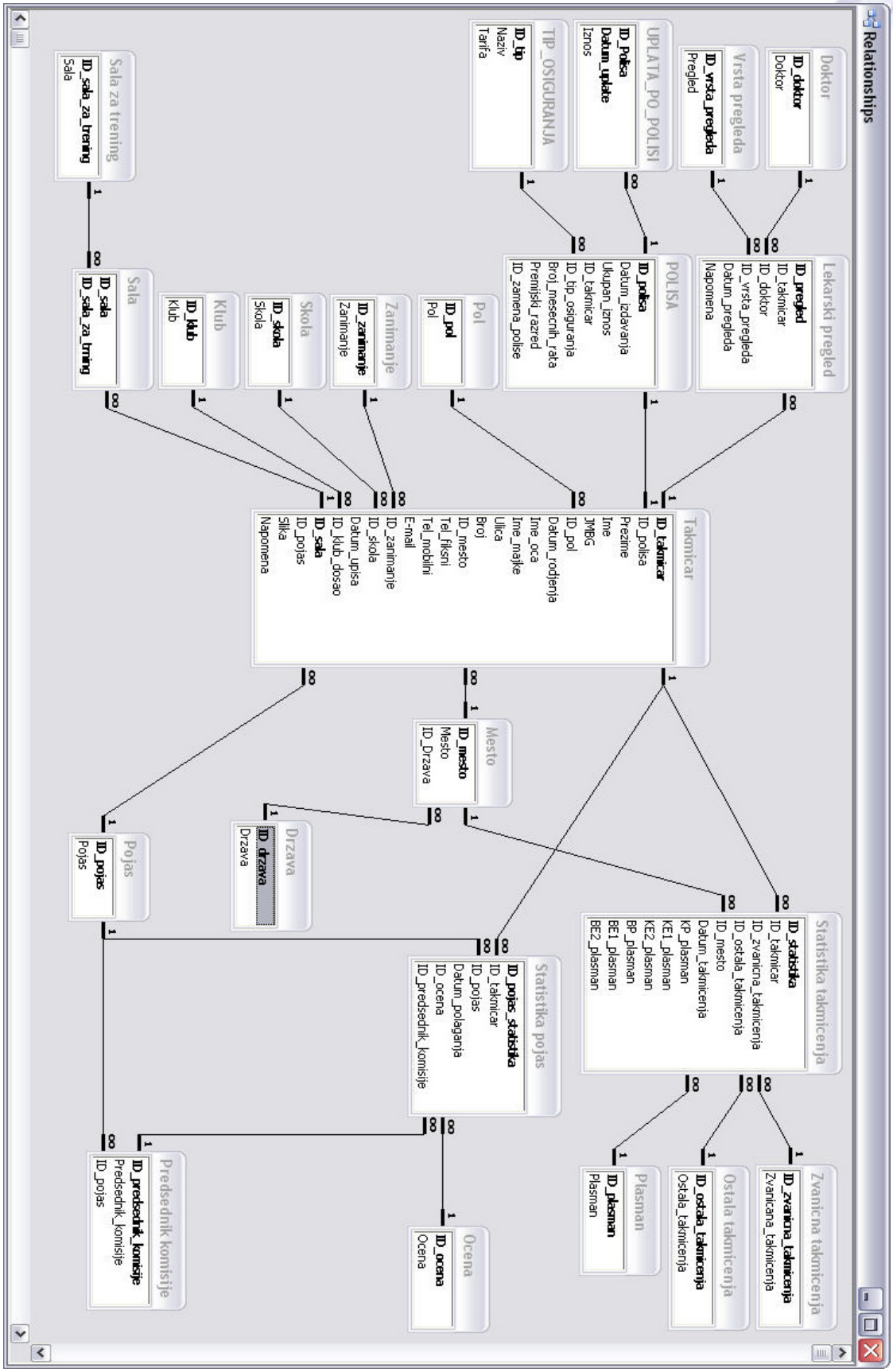
Pod brojem 5 na slici je označen taster **Join Type**.... Klikom na taster ili prečicom **ALT+J** otvara se dijalog sa tri opcije:



Slika 39: Izbor Join Properties

Prva opcija će dozvoliti pojavljivanje takmičara u tabeli **Statistika** samo za one koji već postoje u tabeli **Statistika**. Biranjem druge opcije u tabeli **Statistika** će moći da se "pojave" svi takmičari. Trećom opcijom bismo dozvolili (slično prvoj) pojavljivanje samo onih koji već postoje u tabeli **Statistika**. Praktično prva i treća opcija nemaju nekog smisla u našem slučaju. Stoga odabiramo drugu opciju i potvrđujem izbor klikom na taster **OK**. Takođe potvrdimo i promene osobina relacija još jednim klikom na taster **OK**.Prateći postupak u kome smo kreirali relaciju između tabela **Takmicari** i **Statistika** gde smo povezali polja **ID_takmicar**, povežimo i ostale tabele.

Nakon obavljene operacije imaćemo izgled relacija kao na sledećoj slici:



Slika 40: Prozor Relationships

Konačno zatvorimo prozor relacija i odgovorimo potvrdno na postavljeno pitanje kojim snimamo naše promene na relacijama tabela.



18 UNOŠENJE PRVIH ZAPISA

Pre nego što pređemo na kreiranje formulara za rad sa podacima, iskoristićemo priliku da unesemo podatke direktno u tabele. Način na koji Access smešta podatke u tabele je naizgled veoma jednostavan, a njegovo razumevanje je veoma važno za kasnije dizajniranje formulara i izveštaja. Od tabela koje smo kreirali otvorimo za unos tabelu Takmicari. Dvoklik na naziv tabele uz pritisnut taster CTRL otvara tabelu u dizajn-pogledu.

Popunimo prvi slog tabele sledećim podacima:

Tabela 13: Unos zapisa u tabelu Takmicar

ID_takmicar	1651
Prezime_ime	Graovac Ognjen
Slika	D:\KK Enpi\Slike\GraovacOgnjen.jpg
JMBG	1404994
Pol	M
God_rodj	1994
Ime_oca	Branko
Ime_majke	Mirjana
Ulica	Šumska 7/3
Mesto	Subotica
Tel	024-554-615
E-mail	graovac@yunord.net
Zanimanje	Učenik
Skola	OŠ Kizur Ištvan

Slika 41: Izgled popunjene tabele

	Dragić Bojan	D:\KK Enpi\Slike\Enpi.jpg		M										
554	Đurđev Jovana	D:\KK Enpi\Slike\DjurdjevJovana.jpg	2701994	Ž	1994									
576	Đurić Boris	D:\KK Enpi\Slike\DjuricBoris.jpg	2806985	M	1985									
563	Farmakovski Tamara	D:\KK Enpi\Slike\FarmakovskiTamara.jpg	0507989	M	1989									
551	Govorušić Biljana	D:\KK Enpi\Slike\GovorusicBiljana.jpg	0711989	Ž	1989									
1651	Graovac Ognjen	D:\KK Enpi\Slike\GraovacOgnjen.jpg	1404994	M	1994	Branko	Mirjana	Šumska 7/3	Subotica	024-554-6	064-185-7	graovac	Učenik	OŠ
1650	Graovac Stefan	D:\KK Enpi\Slike\GraovacStefan.jpg	1404994	M	1994	Branko	Mirjana	Šumska 7/3	Subotica	024-554-6	064-185-7	graovac	Učenik	OŠ
569	Hajčunk Tihomir	D:\KK Enpi\Slike\HajcunkTihomir.jpg	2607990	M	1990								Učenik	
2368	Karaman Nikola	D:\KK Enpi\Slike\KaramanNikola.jpg	2709994	M	1994				Subotica				Učenik	OŠ
2370	Klačar Nikola	D:\KK Enpi\Slike\KlacarNikola.jpg	1506994	M	1994								Učenik	OŠ

Otvorimo zatim tabelu **Takmicenja** i unesimo, takođe, jedan slog:

Tabela 14: Unos zapisa u tabelu Takmičenja

ID_takmicenja	1
Takmicenje	Prvenstvo Srbije

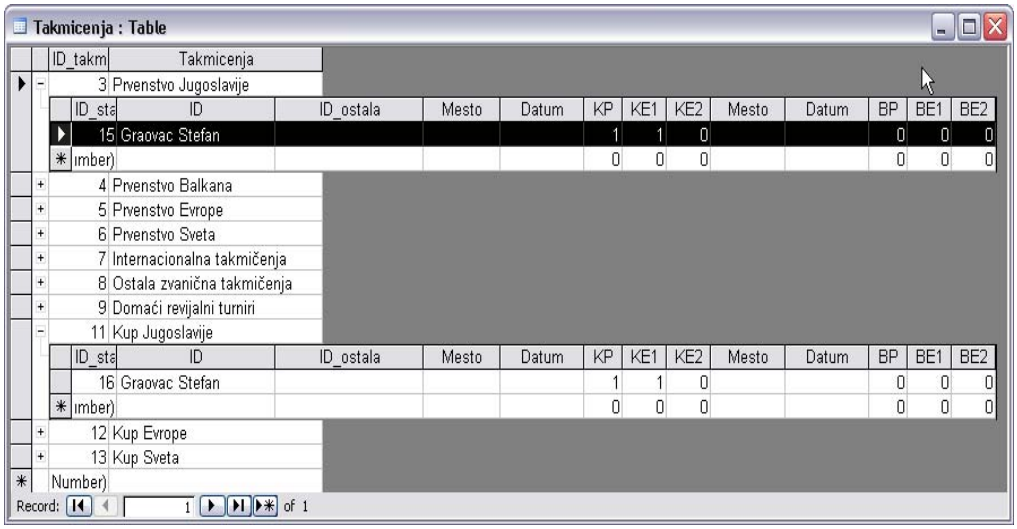
Pošto postoji barem jedan slog u obe tabele možemo otvoriti i treću tabelu - **Statistika**, kako bismo takmičaru evidentirali takmičenje. U polje **ID_Statistika** ne upisujemo ništa, uostalom Access to neće dozvoliti, jer smo to polje definisali kao **AutoNumber**. To polje Access sam popunjava inkrementirajućim brojem. Dakle, popunimo prvi slog tabele **Statistika** sledećim vrednostima:

Tabela 14: Unos zapisa u tabelu Statistika takmičenja

ID_takmicar	(polje "drop-down") biramo takmičara
ID_takmicenja	(polje "drop-down") biramo zvanično takmičenje
Ostakmicenja	(polje "drop-down") biramo naziv takmičenja
MestoK	Npr. Novi Sad
DatumK	27.01.2007
KP	1
KE	0

U trenutku kada započinjemo unos u tabele **Takmicari** i **Takmicenja**, primetimo znak + na levoj strani sloga. Klikom na taj znak Access prikazuje kaskadnu tabelu za koju je direktno vezana tabela sa kojom operišemo, odnosno prikazuje slog ili grupu slogova koji su u relaciji sa slogom otvorene tabele. Za tabelu **Takmicari** klikom na znak + dobija se tabela **Takmicenja** gde su slogovi u međusobnoj direktnoj relaciji:

Slika 42: Dokaz da su dve tabele u relaciji



Sa slike možemo videti da je takmičar sa šifrom **1650** učestovao na nekoliko takmičenja i ostvario zapažene rezultate. Ovo je zgodan način za testiranje ispravnosti relacija i tabela pre nego što se pređe na dizajniranje ostalih elemenata baze, naročito kada operišemo sa više tabela.



19 KREIRANJE ELEKTRONSKIH OBRAZACA - FORMI

Forme, maske, elektronski obrasci ili formulari su neki od naziva koji su odomaćeni za englesku reč *Forms*. Elektronski obrasci predstavljaju ekvivalent papirnim obrascima, a koriste se za unos podataka u elektronske, računaru razumljive, baze podataka. Prednost elektronskih obrazaca nad papirnim je višestruka: unos podataka može biti delimično kontrolisan, u određenim poljima se mogu unapred ponuditi vrednosti (recimo tekućim datumom) dok se pojedina polja mogu popuniti automatski proračunom jedne ili više unetih vrednosti u druga polja.

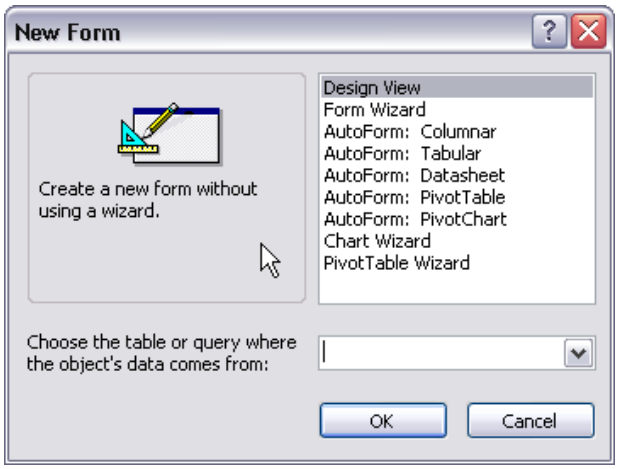
Pre osmišljavanja izgleda elektronskih obrazaca (u daljem tekstu: formi) valja uzeti na uvid papirne obrasce sa kojima su operateri do sada radili. U principu, nama je svejedno kako ćemo ih dizajnirati dok god poštujemo određena pravila, ali na taj način bismo olakšali operaterima prilagođavanja na nov način rada.

Osnovna pravila kojih se treba držati su da se ne koriste blještave boje i ukrasni fontovi. Naše kreativne sposobnosti za ukrašavanje ćemo ostaviti za neko drugo mesto. Operateru koji često provodi i više godina na istom radnom mestu će biti vrlo zamorno da gleda u šaren ekran i da čita kitnjasta slova po ceo dan. Raspored polja koji se popunjava mora biti logičan i ravnomerno uravnat. Takođe valja predvideti da se kompletan unos može obaviti sa tastature, dakle, BEZ korišćenja miša. Kada se jednom usvoji ova navika unos može predstavljati pravo zadovoljstvo uz veliku uštedu u vremenu.

Za operativne sisteme Windows familije postoji ogroman broj programa u kojima se pojavljuju određene forme.

19.1 Prva forma – Talmicari

Pod pretpostavkom da je tabela **Takmicari** dobro organizovana, možemo preći na kreiranje prve forme. Pređimo na karticu **Forms**. U prozoru baze postoje tri tastera: **Open** (otvara za rad postojeću formu), **Design** (otvara za izmenu dizajna postojeću formu) i **New** (kreiranje nove forme). Biramo taster **New** i dobijamo sledeći izbor:

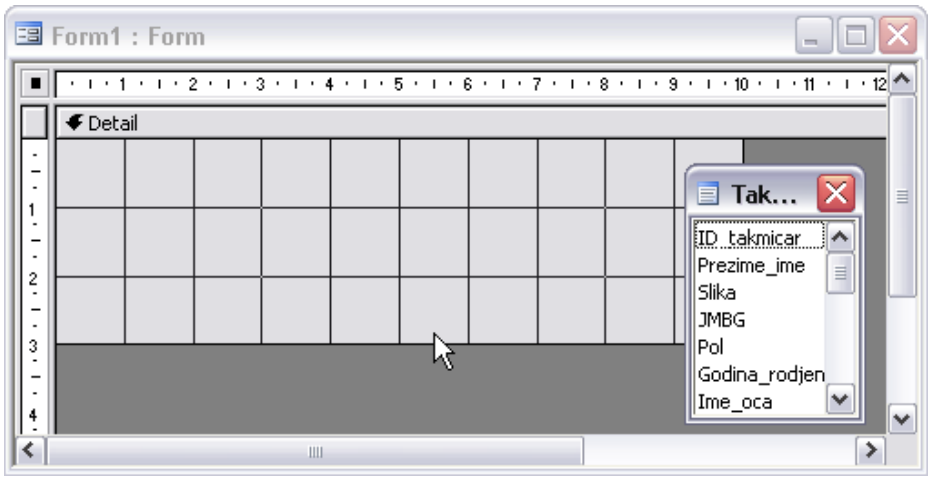


Slika 43: Izbor opcije za kreiranje nove forme

Od ponuđenih opcija prvog koraka dat je izbor (sa značenjem):

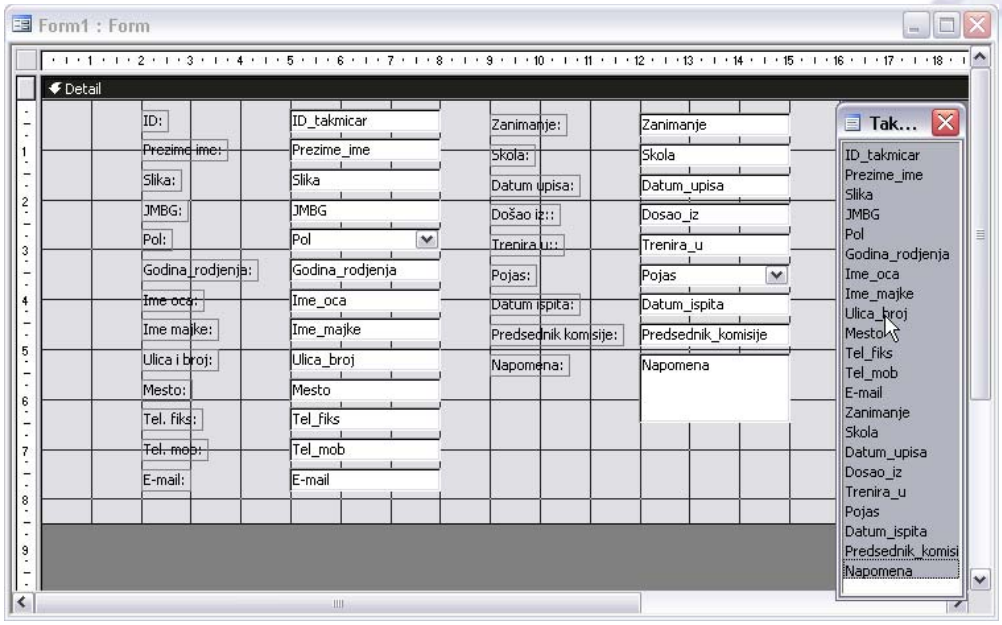
- **Design View** - dizajniranje forme "od nule" bez upotrebe čarobnjaka; sve opcije moramo samo podesiti;
- **Form Wizard** - čarobnjak koji na osnovu odgovora na postavljena pitanja deo opcija postavlja automatski, te sam raspoređuje polja za unos;
- **AutoForm**: (**Columnar**, **Tabular**, **Datasheet**) kreira formu bez ikakvih pitanja na osnovu tabele koju odaberemo; podopcije predstavljaju način na koji će polja za unos biti raspoređena na formu;
- **Chart Wizard** - čarobnjak koji kreira formu sa grafikonom na njoj (koristi se uslugama Microsoft Excela koji pri tome mora biti instaliran);
- **PivotTable Wizard** - čarobnjak koji iz podataka iz tabele kreira izvedenu tabelu (takođe zahteva Microsoft Excel);

Markiramo opciju **Design View** klikom miša i otvaramo padajuću listu za izbor tabele u koju će se podaci upisivati (**Takmicari**), odnosno na osnovu koje će se formirati polja za unos. Klikom na taster **OK** otvara se prazna forma i prozor sa nazivima polja koja se nalaze u tabeli **Takmicari**:



Slika 44: Design View za formu takmičari

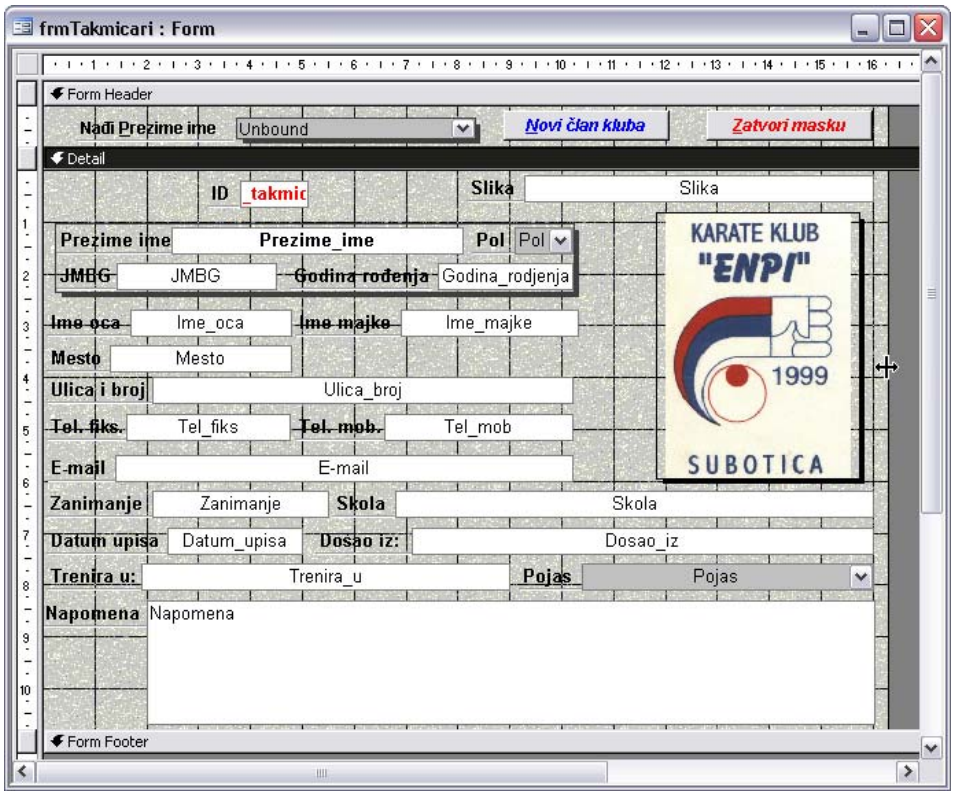
U prozoru sa listom polja kliknimo prvo na **ID_takmicar** i uz pritisnut taster **SHIFT** na **Napomena**. Ovim selektujemo sva polja iz tabele **Takmicari** koju ćemo zatim prevući na prazan prostor forme, odnosno ispod odeljka **Details**:



Slika 45: raspored polja u pogledu Design View tabele takmičari

Primetimo da se u prvoj koloni nalazi komentar identičan nazivu polja tabele, a da se u drugoj koloni nalaze polje za unos. Takođe, pri ovoj operaciji Access sam postavlja veličinu polja za unos na istu vrednost. Prozor sa spiskom polja (**Field List**) sada možemo zatvoriti.

Prvo što ćemo uraditi jeste da promenimo veličinu polja za unos. Kliknimo na formu u prazan prostor u odeljku **Details** kako bismo deselektovali izbor polja.



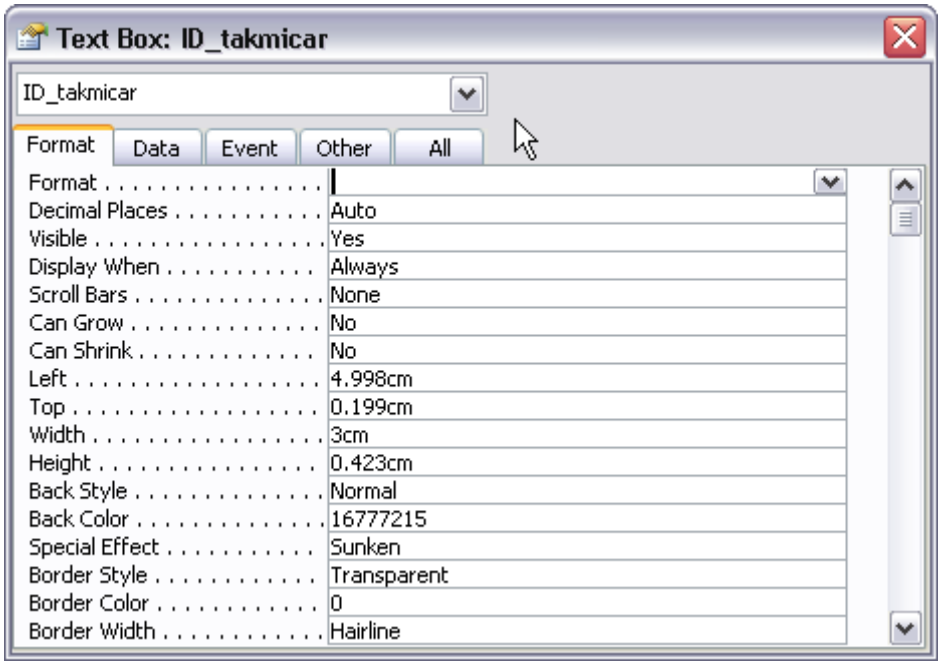
Slika 46: Dodatno uređena forma za unos takmičara u bazu podataka

Klikom desnim tasterom na prvo polje - **ID_takmicar** otvara se skraćeni meni, sa kojeg biramo poslednju opciju **Properties** = Osobine.

Prozor sa osobinama polja ima pet kartica na kojima su grupisane osobine polja. S leva na desno nazivi i sadržaji kartica su sledeći:

- **Format** - način na koji će na ekranu biti prikazano polja i koje će osobine uzeti u smislu oblika, veličine, vrste fonta, boje, itd; posebne osobine na ovoj kartici su:
  - **Visible** - polje će biti vidljivo ili ne
  - **Display When** - polje će biti vidljivo na ekranu, na odštampanom papiru ili na oba
  - **Scroll Bars** - vertikalni/horizontalni klizač postoji ili ne
  - **Can Grow/Can Shrink** - u zavisnosti od sadržaja polje se može/ne može automatski povećati/smanjiti
- **Data** - na ovoj kartici se podešava izvor podataka, mogućnost njegove izmene, kontrola unosa podataka u polje, osnovna vrednost i način ponašanja filtera nad poljem
- **Event** - na svakoj operaciji nad poljem aktivira se određeni događaj (**Event**) koji može pokrenuti program; u narednim lekcijama ćemo neke od događaja iskoristiti za automatizovanje rada
- **Other** - ostale osobine polja koje ne spadaju niti u jednu od preostale četiri grupe; neke od osobina ćemo koristiti u našoj bazi
- **All** - sve osobine sa ostale četiri kartice se nalaze na ovoj kartici.

Vratimo se na karticu **Format** i u osobinu **Width** polja **ID_takmicar** upišimo vrednost **3cm**. Za sva ostala polja i labele postupak definisanja veličine i naziva je isti, stoga ove promene izvedimo do kraja po sopstvenim afinitetima. Na kraju snimimo formu (**File/Save As**...) pod imenom **frmTakmicari**.



Slika 47: Properties polja ID_takmicar

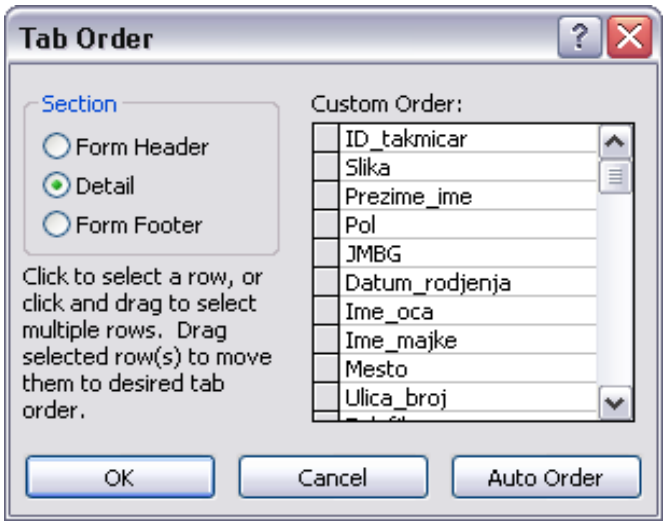


Jednostavna forma kakvu smo dobili može poslužiti za unos, izmenu i brisanje zapisa tabele **Takmicari**.

Formu **frmTakmicari** otvorite u pogledu za izmenu dizajna i otvorite meni **View** i opciju **Tab Order**.... Ovde se može podesiti redosled po kome će se kurzor kretati kada korisnik pritisne taster **Tab** ili **Enter**. Smisleno je da se kurzor premesti na sledeće polje, a ne da se podaci popunjavaju odozdo na gore ili nekim drugim redosledom. Obratite pažnju na spisak polja i zatim pritisnite taster **Auto Order**. Access će pokušati da poređa polja logičnim redosledom, a ako ne primetite razliku pre apliciranja ove operacije i posle nje, to znači da je redosled dobar.

Obratimo pažnju, Access ponekad zna da omane: ako su polja u dve ili više kolona, rezultat će biti neizvesan, u zavisnosti od naše namere - da li će se popunjavati prva kolona, pa zatim druga ili naizmenično obe.

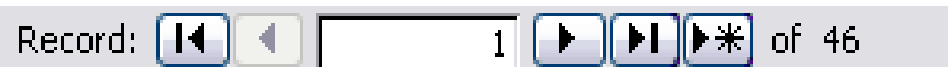
Slika 48: Podešavanje rasporeda za popunjavanje podataka



19.2 Kontrole u formi

Svaka novokreirana forma sadrži standardne kontrole: kontrole za navigaciju i selektovanje sloga. Kontrolama za navigaciju korisnik se kreće kroz slogove napred i nazad; skače na prvi i poslednji ili se pozicionira na novi slog za novi unos:

Slika 49: Kontrolama za navigaciju



Ova slika je preuzeta iz forme **Takmicari**. Sleva na desno, u ovom navigacionom baru, stoji:



- **Record**: - slog ili zapis
- Taster za skok na prvi slog tabele
- Taster za skok na prethodni slog tabele
- Polje sa relativnim numeričkim indikatorom tekućeg sloga tabele
- Taster za skok na sledeći slog tabele
- Taster za skok na poslednji slog tabele
- Taster za skok na novi slog
- **of 46** - ukupan broj slogova, odnosno zapisa

U polje sa indikatorom tekućeg sloga se može ukucati broj i nakon pritiska na **Enter** forma će prikazati redni broj sloga koji ste uneli u ovo polje.

NAPOMENA:

Sa tastature, prečica CTRL+A selektuje sve slogove!

Alternativne kontrole se mogu realizovati na više načina - mogu biti u obliku tastera, površine, hiperlinkova ili praktično bilo čega drugog (fotografije na primer). Najčešće se koriste kontrole u obliku tastera na kome je napisano nešto poput **Novi član kluba**. Sada ćemo ukloniti standardne kontrole i postaviti tastere za navigaciju.

Otvorimo formu u dizajn pogledu (selektujmo formu **frmTakmicari** i kliknimo na taster **Design**). Desnim tasterom kliknimo na kvadratić u levom gornjem uglu forme i selektujmo **Properties**. Pređimo na karticu **Format** i postavimo sledeće osobine i vrednosti:

Slika 50: Selektujmo celu formu



Tabela 14: Kontrole u formi Takmicari

Scroll Bar	Neither	Klizači se nikada neće videti
Record Selectors	No	Bez selektora sloga
Navigation Buttons	No	Kontrole za navigaciju se neće videti
Auto Center	Yes	Forma se po otvaranju pojavljuje na sredini ekrana
Border Style	Dialog	Veličina forme ne može da se menja
Caption	Takmičari	Upišimo u naslovnu liniju forme tekst "Takmičari"

Otvorimo formu **frmtakmicari** u modu za izmenu dizajna. Proverimo da li na ekranu postoji toolbar (linije sa ikonama) pod nazivom **Toolbox**. Ako ne postoji, kliknimo desnim tasterom miša na bilo koju ikonu na postojećem toolbaru i u listi koja se pojavi uključimo **Toolbox**.



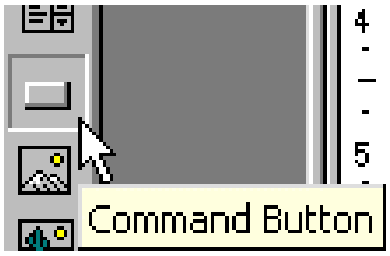
Slika 51: Paleta ikona toolbar

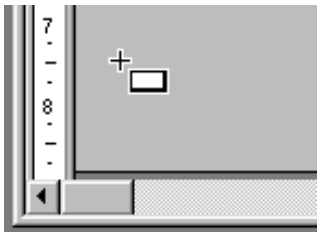


U ovom toolbaru se nalaze ikone koje predstavljaju kontrole koje se mogu postaviti na aktivnu formu. Upotreba miša i ovakvih ikona znatno skraćuje postupak dizarniranja, te je njihova upotreba ovde poželjna.

Kliknimo mišem na ikonu tastera iz toolbara i zatim na prazno mesto na formi u njenom donjem delu:

Slika 52: Izbor opcije toolbara i nanošenje na formu





Otvara se prozor jednog od čarobnjaka, koji će umesto vas postaviti odgovarajući VBA (Visual Basic for Applications) program i sliku odnosno tekst na taster. Rad sa ovim čarobnjakom je prilično trivijalan, mi ćemo krenuti drugim putem i preskočiti čarobnjaka kako bi se upoznali sa VBA. Na ekranu koji se pojavio pritisnimo taster **Cancel**. Sada smo dobili formu sa jednim nefunkcionalnim tasterom na kome piše nešto poput **Command1**.

19.3 Događaji (eng. Events)

Svaka korisnička akcija nad formom signalizira da se aktivira određeni program vezan za događaj, ako takav uopšte postoji. Događaja ima dosta i oni pokrivaju sve moguće aspekta rada sa objektima u Accessu. Za primer, pri otvaranju forme aktiviraju se tri događaja: **OnOpen**, **Onload** i **OnActivate**, dok se četvrti događaj aktivira i po učitavanju forme i pri davanju fokusa formi: **OnGotFocus**. Za svaki od tih događaja u odgovarajući rubriku kartice **Properties** možemo postaviti program.

Princip rada je sledeći: kada kliknemo mišem na određeni taster na formi, Access provera da li postoji VBA program/makro za taj događaj i izvršava ga. Ukoliko ne postoji, ne dešava se ništa. U našem slučaj kada pređemo u mod za rad sa formom **Takmicari** na kojoj je taster **Command1**, levi klik mišem na taster neće inicirati nijedan program - jer on još nije napisan.

Ovom tasteru ćemo promeniti naziv u **Zatvori masku**; napravićemo da se klik na taster može proizvesti sa tastature (**ALT+Z**) i napisati vrlo jednostavan program koji će zatvoriti formu.

U dizajn modu forme **Takmicari** kliknimo desnim tasterom miša na taster **Command1** i otvorimo karticu osobina (**Properties**). Pređimo na tab **All** i popunimo polje **Caption** sledećim sadržajem:

&Zatvori masku



Konačno, pritisnimo taster **Enter** sa tastature. Primetimo da se naziv tastera na formi promenio iz **Command1** u **Zatvori masku**, a slovo **Z** je podvučeno. Znak **&** (tzv. ampersend) ispred određenog karaktera označava prečicu sa tastature za pristup tom tasteru u kombinaciji sa tasterom **ALT**. To znači da korisnik može pristupiti ovom tasteru i ostvariti klik na njega bilo gde da se kurzor nalazi pritiskom kombinacije **ALT+Z**. Dok je forma u fokusu, nije neophodno kombinovati hot-key sa tasterom **ALT**. **Za pretpostavku**, ako se nekom od menija pristupa preko tastera sa **&z** i ako je u isto vreme aktivna forma koja na sebi ima taster sa **&z**, meniju ćemo pristupiti sa **ALT+z**, a tasteru na formi samo pritiskom na **z**. Valja obratiti pažnju da se na formi ne postavi isti hot-key za pristup dvoma komandnim tasterima: pri aktiviranju "vrućeg tastera" aktiviraće se prvi na formi.

Ovom tasteru na formi smo dodelili prečicu na tastaturi; potrebno je još i da postavimo odgovarajući program, što ćemo učini na jedan od dva načina.

19.4 Makro program

Pređimo na karticu makroa i kliknimo na taster **New**. Otvara se prozor u kome pišemo makro. U prvo polje, pod **Action** upišimo **Close**. Polja **Object Type** i **Object Name** ostavimo prazna. Pod **Save** upišimo (ili ako je već upisano, ostavimo) reč **Prompt**). Zatvorimo makro i pri tome ga snimimo pod imenom **Zatvori**.

Ovim smo napravili makro koji, kada se aktivira, zatvara aktivnu formu (onu koja je u fokusu). Da smo u **Object Type** i **Object Name** upisali naziv neke druge forme koja je otvorena, mogli bi da je zatvorimo bez obzira da li ima fokus ili ne, da li je skrivena ili vidljiva.

Vratimo se na formu **Takmicari** i otvorimo je u pogledu za izmenu dizajna. Kliknimo na novokreirani taster i otvorimo karticu osobina (**Properties**). Pređimo na tab **Event** i pronađimo u listi događaj **On Click**. Ako postavimo kurzor u to polje, videćemo da ja polje padajućeg tipa sa strelicom na dole. Kliknimo na strelicu i iz liste u kojoj postoje dve stavke (**[Event procedure]** i **Zatvori**) odaberimo stavku **Zatvori**. Ovim smo tasteru pridružili makro koji se aktivira klikom miša na njega. Sem na klik miša, ovaj taster se može "pritisnuti" na više načina: prečicom **ALT+Z** sa tastature; davanjem fokusa tasteru i pritiskom na **ENTER** ili razmaknicu (spacebar).

Pogledajmo kako taster radi: pređimo u pogled za rad sa podacima i kliknimo na taster. Dobićemo upozorenje da je promenjen izgled i način rada forme, te da izmene nisu snimljene. Snimimo ih pritiskom na taster **Yes**. Konačno otvorimo (dvoklikom miša) formu ponovo i kliknimo na taster **Zatvori formu**. Forma se zatvara i sa ovim tasterom smo završili rad što se makroa tiče.

Bolje koristiti VBA umesto makroa, ma kako oni izgledali lakši za pisanje. Makroi u verziji Accessa 2000 i novijim postoje samo zbog kompatibilnosti sa ranijim verzijama; lako je moguće da ih već u narednoj verziji neće biti uopšte.



19.5 VBA program

Ukoliko smo se odlučili da umesto makroa koristimo procedure jezika VBA (ovde bi reč "preporuka" trebalo da bude zamenjena sa rečju "obaveza"), postupićemo na sledeći način. Otvorimo formu u dizajn pogledu, selektujmo taster **Zatvori formu** i na kartici **Event** osobina pronađimo za njega događaj **On Click**. Otvorimo padajuću listu i iz nje odaberimo prvu opciju - **[Event Procedure]**. Na desnoj strani tog polja pojavljuje se mali taster sa tri tačke na njemu. Kliknimo na njega. Otvara se prozor Visual Basic Editora u kome pišemo odgovarajuće VBA programe.

Na ekranu ćete dobiti sledeće:

```
Option Compare Database
------------------------------

Private Sub Command1_Click()

End Sub
```

"Private Sub Command1_Click()" ovde znači da podproceduru koristi samo aktivna forma, da se ona aktivira na klik (Click) na naš taster 'Zatvori formu' (Command1). Sav program za ovaj događaj se smešta između klauzula 'Sub' i 'End Sub'.

Pozicionirajmo se između klauzula **Private**... i **End Sub** i pritisnimo jedanput taster **Tab**. Uvlačenje se koristi kako bi program bio čitljiviji i uredniji. Različita uvlačenja koristimo za odvojene delove podprograma. Dva-tri razmaka su dovoljna da bi se stekao pregled gde celina počinje i gde se završava.

Upišimo ovde sledeće:

```
DoCmd.Close
```

Primetićemo da se posle tačke iza **DoCmd** (što znači "Izvrši komandu") otvara lista sa mogućim naredbama koje mogu slediti. Sve drugo, osim toga što je u listi, neće imati nikakvu funkciju i program će prijaviti grešku.

Zatvorimo prozor VBA, sninimo formu i pređimo u pogled za rad sa podacima. Klinimo na taster **Zatvori formu** i forma će se zatvoriti.

Poštujući prethodno opisanu proceduru, dodajmo još jedan taster na formu i promenimo mu naziv u **&Novi član kluba**. Za ovaj taster, na događaj **On Click** postavimo sledeći program:

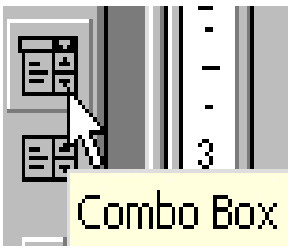
```
DoCmd.GoToRecord acActiveDataObject, , _
acNewRec ID_takmicar.SetFocus
```

Prvi red zahteva da se pređe na novi slog aktivne forme, dok se u drugom redu fokus premešta sa taster **Novi član kluba** na polje **ID_takmicar**. Taster **Novi član kluba** se može aktivirati i sa tastature kombinacijom tastera **ALT+N**.



19.6 Pretraga u formi

Polje za pretragu se obično postavlja kao Combo polje, tj. kao padajuća lista sa mogućnošću direktnog unosa ili uz izbor iz ograničene liste podataka. Kucanjem teksta ili brojeva u ovo polje, Access pokušava da dopuni ostatak sadržaja kojeg korisnik traži. Zašto Combo polje, a ne obično? Jednostavno zato što ono ima neke osobine koje obično polje nema (opciju **Limit to List** ili **On Not In List**). Postavimo jedno polje za pretragu takmičara po ključu **ID_takmicar**.

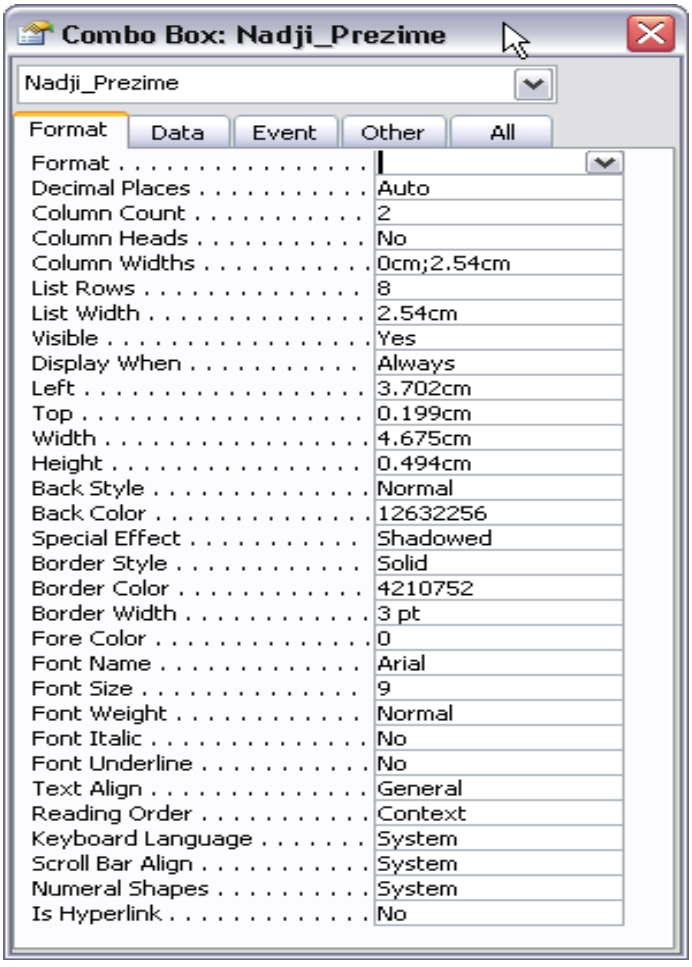


Slika 53: Izbot za Combo Box polje

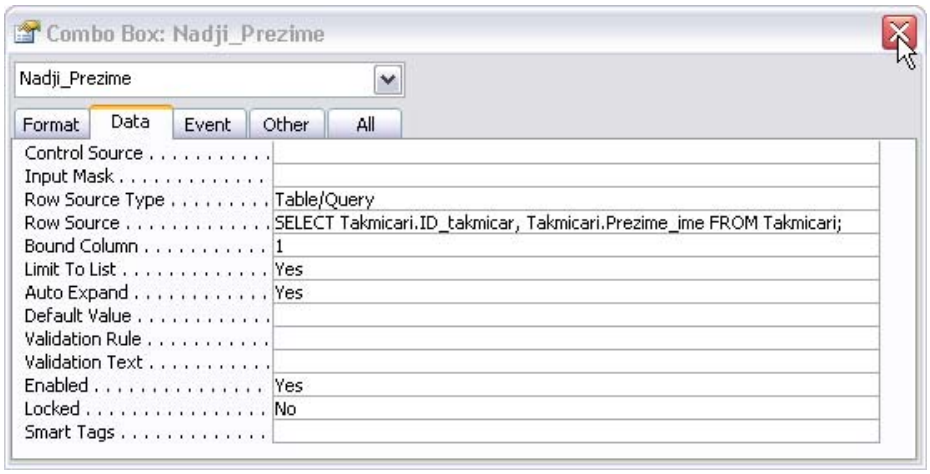
Prvo kliknimo na ikonu Combo polja i zatim ga prenesimo na formu.

Ako imamo instalirane "čarobnjake", pokrenuće se jedan od njih koji će nas provesti kroz proceduru postavljanja parametara za Combo polje na osnovu odgovora na ponuđeni izbor. Ovu proceduru ćemo preskočiti, kako bismo mogli da vidimo šta se iza nje krije.

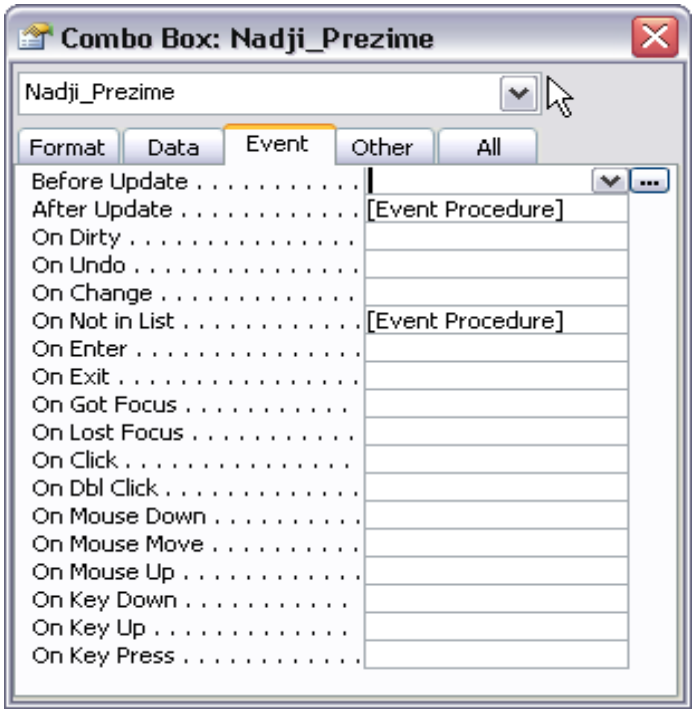
Pritiskamo taster **Cancel**, otvaramo karticu osobina za Combo polje i postavljamo sledeće osobine i vrednosti:



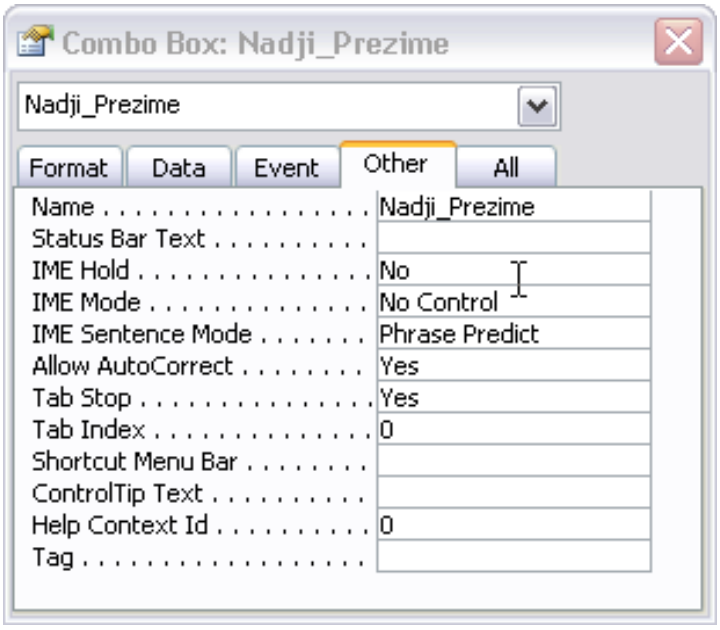
Slika 54: Podešavanje parametara za Combo Box polje kartice Format



Slika 55: Podešavanje parametara za Combo Box polje kartice Data



Slika 56: Podešavanje parametara za Combo Box polje kartice Event



Slika 57: Podešavanje parametara za Combo Box polje kartice Other



Zatim kliknimo na Combo polje (**Nadji_Prezime**) i predvidimo događanja:

- ako unesemo prezime takmičara koja postoji, forma prikazuje traženi zapis, briše se sadržaj polja **Nadji_Prezime**;
- ako unesemo prezime koja ne postoji ili unesemo šifru u ovo polje, izbor se poništava, presreće se poruka o grešci koju Access generiše, izdaje se poruka o grešci i fokus se vraća na Combo polje **Nadji_Prezime.**

Da bismo ispunili ove uslove, koristićemo program sa dva događaja za ovo polje. Za prvi uslov odgovarajući događaj je **After Update** (posle izmene), dok se za aktiviranje programa po drugom događaju koristi **Not In List** (nije u spisku).

U događaj **AfterUpdate** upišite sledeći program:

```
Dim rs As Object

Private Sub Nadji_Prezime_AfterUpdate()

Dim rs As Object

Set rs = Me.Recordset.Clone
rs.FindFirst "[ID_takmicar] = " & Str(Me![Nadji_Prezime])
Me.Bookmark = rs.Bookmark
Nadji_Prezime.SetFocus
Nadji_Prezime.Value = ""

End Sub
```

Da bi Access znao sa čime operiše, to mu se mora reći. Objekat **rs** se definiše kao **Recordset** - skup zapisa. Nad tim skupom zapisa se sprovodi pretraga (**rs.FindFirst**) i zatim se pokazivač premešta na nađeni slog (**Me.Bookmark**). Fokus se premešta na **ID_takmicar** dok se sadržaj Combo polja briše.

U događaj **NotInList** upišite:

```
Private Sub Nadji_prezime_NotInList(NewData As _
   String, Response As Integer)

On Error GoTo Greska

Greska:
Nadji_Prezime.Undo
Response = accDataErrorContinue
MsgBox "Neispravan unos ili takav Član na postoji!", _
       vbInformation,  "G R E Š K A"

End Sub
```



Klauzula **On Error GoTo Greska** šalje program na kodnu labelu **Greska**: i izvršava program iza nje. Prvo se poništava unos sa **Undo**, presreće se greška koju Access generiše sa **accDataErrorContinue** i izdaje se poruka o grešci **MsgBox**.

Proceduru opisanu do sada možemo jednako primeniti i na ostale forme uz neznatne izmene, kao i postupak kreiranja polja za pretragu.

19.7 Samostalno padajuća lista

Sitnih poboljšanja nikad dosta: ako isprogramirate događaje na elementima forme u Accessu, unos podataka može biti jako olakšan operateru. Evo i jednog primera.

Upotrebom procedura događaja u poljima obrasca, možemo vrlo precizno definisati ponašanje naše aplikacije u Accessu, pa sa neznatno malo truda možemo obezbediti korisnikovu veću produktivnost. Evo kratkog primera: ukoliko smo u obrazac postavili padajuću listu (dropdown combo box) i pripremili je inicijalnim vrednostima, možemo definisati da se ona otvori čim primi fokus. Da bismo to uradili, unesimo sledeći kod u proceduru događaja (najpre desni klik na padajuću listu, odaberimo Build Event, pa Code Builder i izaberite proceduru GotFocus):

```
Private Sub obr1Lista_GotFocus()

Me![obr1Lista].Dropdown

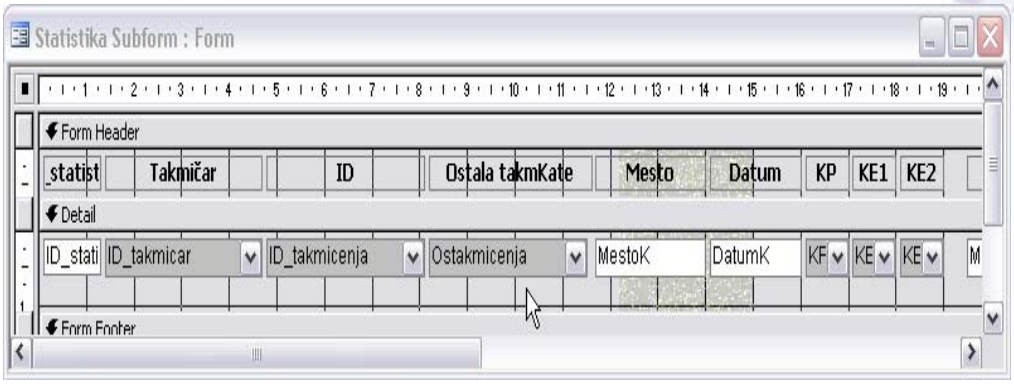
End Sub
```

U ovom primeru, prilikom formiranja padajuće liste, dato joj je ime obr1Lista; Code Builder će se sam pobrinuti da procedura dobije ime vezano za pravo ime liste. I to je sve: kasnije, prilikom unosa podataka u obrazac, lista će se otvoriti, a operater unosa će strelicama nadole i nagore odabrati željenu vrednost i potvrditi je pritiskom na ENTER. Da nije primenjena ova tehnika, operater bi morao da pritisne ALT+DOWN pre pregleda liste, što posle mnogo ponavljanja u toku dugotrajnog unosa može da bude vrlo zamarajuće.

19.8 Podforma (podmaska ili podobrazac)

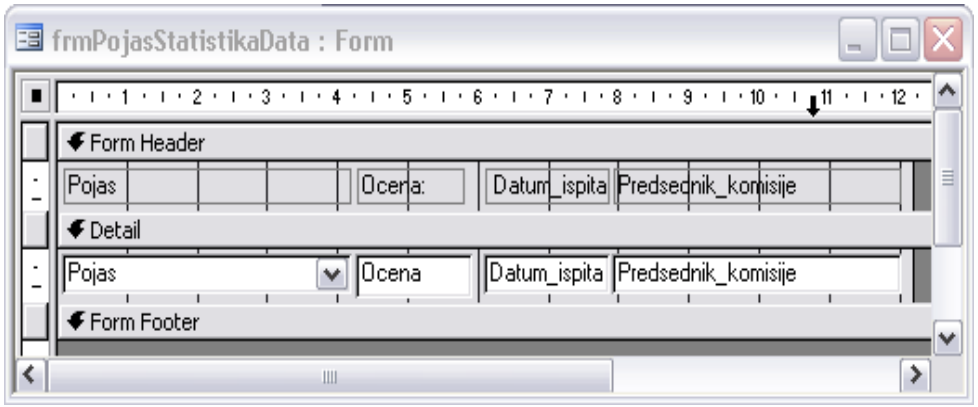
Podforme su klasične forme koje se "ugnežđuju" u glavnu formu (**Master**) i vezane su sa njom preko određenog kriterijuma. Obično služe za prikazivanje dodatnih podataka o nekom pojmu koji se šturo pojavljuje kroz šifru na glavnoj formi. Na našem glavnom modulu se pojavljuju dve podforme.

Podforma se isprva kreira kao obična forma i zatim se postavlja na glavnu formu. Kreirajmo jednu praznu formu, za **Record Source** postavimo tabelu **Statistika** i sva polja prevucimo iz liste polja (**Field List**) na formu. Selektujmo sva polja i postavimo osobinu **Enabled = False.** Podforma za statistiku će sada imati izgled sličan izgledu na slici:



Slika 58: Podforma Statistika takmičenja

Nju snimimo pod imenom **StatistikaSubform**. Kreirajmo zatim i podformu za pojaseve po istom postupku, s tim da će **Record Source** u ovom slučaju pokazivati na tabelu **PojasStatistika.**

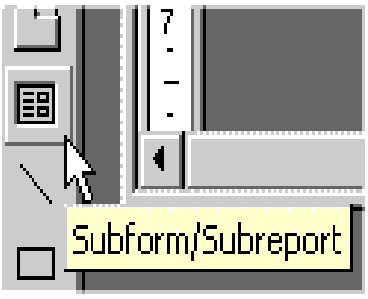


Slika 59: Podforma pojas

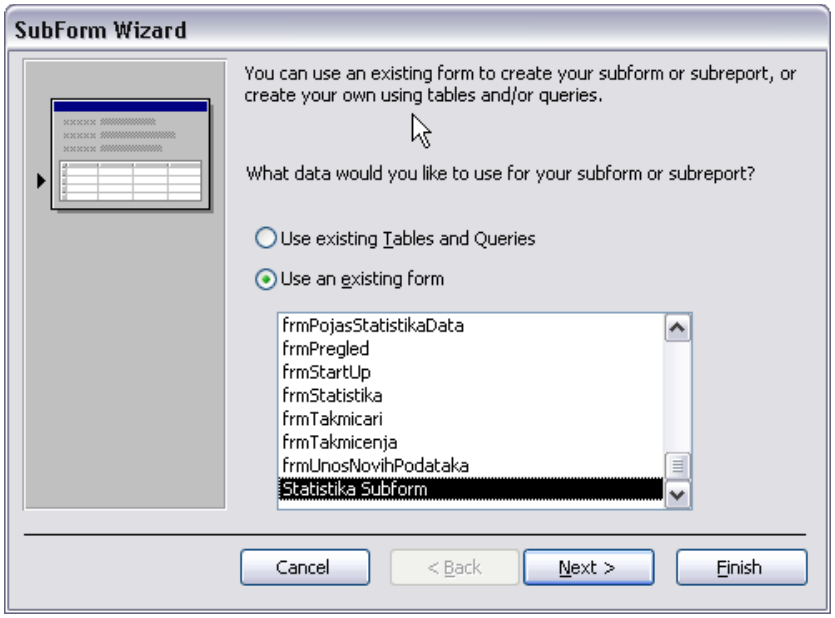
19.9 Forma sa podformom

Vratimo se na dizajn izvršnog modula.

Iz palete alatki **Tools** pronađimo taster za umetanje podformi (ili podizveštaja - i izveštaji mogu imati podizveštaj) i kliknimo ispod podataka na glavnoj formi, na mesto gde želimo da postavimo podformu. Iz liste koja će nam pomoći da odaberemo podformu koju postavljamo odaberimo postojeći **StatistikaSubform**:



Slika 60: Taster za umetanje podformi

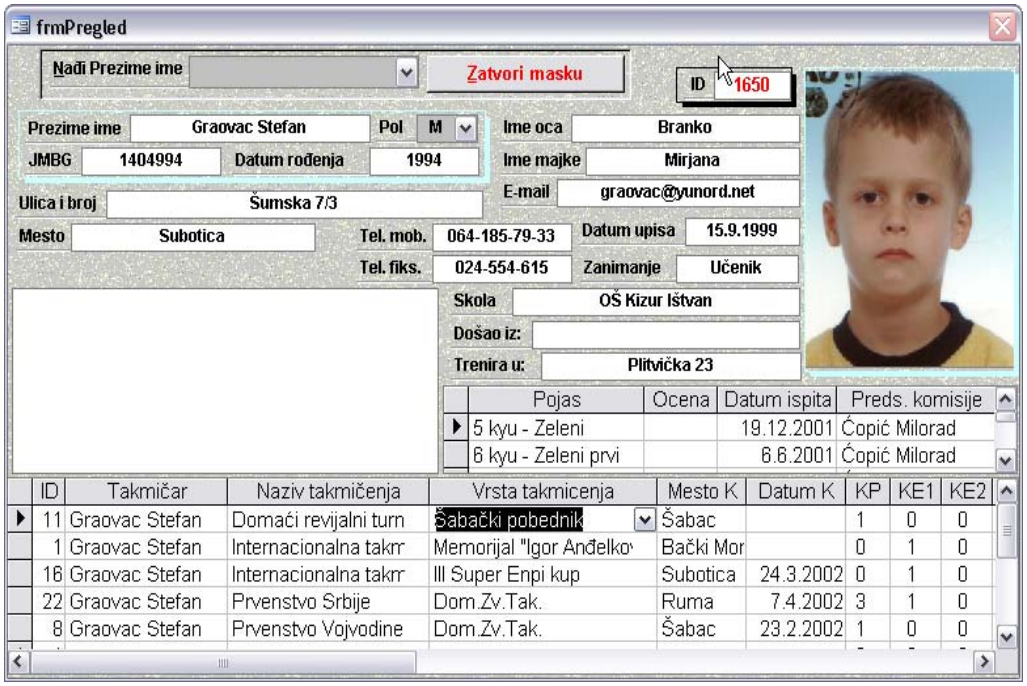


Slika 61: Povezivanje forme i podforme

Na sledećem ekranu imamo mogućnost da definišimo preko kog podatka će se biti povezane glavna forma i podforma. Kliknimo na opciju **Define My Own** i postavimo vrednosti biranjem iz padajućih lista.

Pritisnimo zatim taster **Finish**. Ovom procedurom smo obezbedili da podforma na osnovu broja takmičenja i položenih ispita za pojaseve prikaže njegove potpune podatke.

Izvršni modul aplikacije će na kraju imati konačan izgled nalik ovome:



Slika 62: Finalni izgled forme Pregled sa dve podforme;

20 KOMANDNA TABLA

Komandna tabla je, u stvari, skup tastera kojima se pristupa drugim elementima programa - formama i izveštajima. Ona može sadržati i servisni deo gde će korisnik arhivirati bazu, uraditi kompresiju i slično.

Što se same komandne table tiče, ona će u našem slučaju biti obična "nevezana" forma (forma koja ne koristi niti jednu tabelu za prikaz podataka) setovana tako da "drži kontrolu" nad aplikacijom.

20.1 Izgled komandne table

Predvidećemo mesto za slike na levoj strani. U desnom delu ćemo postaviti tastere za pristup formama i izveštajima baze. U našem primeru, svi tasteri neće se nalaziti na jednoj komandnoj tabli, pošto je aplikacija složenija, neke od tastera će mo postaviti na drugi ekran komandne table, naročito one tastere koji se ređe koriste (kao što su razni izveštaji ili unos podataka).



Slika 63: Komandna tabela aplikacije



Slika 64: Properties komandne table



Formu ćemo postaviti na fiksnu veličinu bez mogućnosti izmene razvlačenjem mišem. Onemogućićemo standardne tastere za **Minimize/Restore/Maximize** i **Close**. Takođe ćemo joj isključiti i sve ostale standardne objekte: selektor zapisa, komande za kretanje kroz slogove i oba klizača.

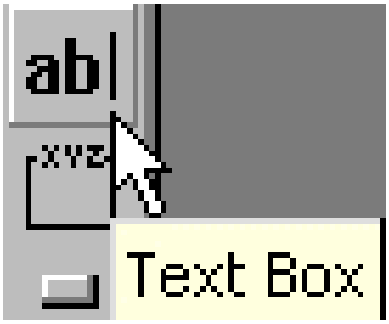
Pređimo na karticu formi i kliknom na taster **New**. Iz izbora odmah selektujmo **Design View** i kliknimo na OK. Otvorimo karticu osobina za formu i postavimo vrednosti opcija kao na slici.

U Accessu otvorimo meni **Insert** i zatim kliknimo na **Picture**....

Pronađimo sliku **koju želimo**, selektujmo je i potvrdimo izbor. Access će zatim kao objekat uvesti sliku na formu.

Kliknimo na ikonu **Text Box**, pa postavimo polje negde pri vrhu forme. Selektujmo labelu i obrišimo je pritiskom na taster **Del**, tako da ostane samo polje. Kliknimo na polje, pritisnimo taster **F2** (editovanje) i u ovo nevezano polje upišimo: **=date()**

Slika 65: Ikona Text Box



Na kartici osobina za ovo polje pronađimo vrednost **Format** i iz padajuće liste odaberimo **Long Date**. Tasteri za brz pristup boji polja i ivice polja se nalaze u grupi tastera pod nazivom **Formatting**.

Slika 66: Dodatno uređenje forme



Za ovo polje postavimo i sledeće osobine:

- **Enabled: No** (ne može uzeti fokus)
- **Locked: Yes** (onemogućuju izmene)
- **Text Align: Center** (centralno uravnat tekst)

Zatim postavimo jedan komandni taster i odgovorimo sa **Cancel** kada se startuje čarobnjak. **Width** i **High** postavimo na vrednosti **4,5cm** i **0,9cm** respektivno. Na novodobijenom tasteru će pisati nešto nalik **Command2**. Dok je taster selektovan, pritisnimo prečicu **CTRL+C** (kopiraj na Clipboard) i zatim **CTRL+V** (vrati sa Clipboarda).



21 FUNKCIJE KOMANDNE TABLE

Komanda tabla, kao centralni deo aplikacije, povezuje ostale elemente baze, odnosno programa.

Otvorimo formu **frmKomandnatabla** u režimu izmene dizajna. Tasterima ćemo prvo dodeliti nazive i prečice sa tastature za brz pristup:

Tabela 15: Opis tastera sa komandne table

Opis	Name	Caption
Pregld članova kluba, položenog pojasa i ostvarenih rezultata	Pregled	&Pregled
Otvara novu komandnu formu sa tasterima za unos podataka	Unos podataka	&Unos podataka
Otvara novu komandnu formu sa tasterima za razne izveštaje	Izvestaji	&Izveštaji
Arhiviranje baze + Kompresija baze	Kompresija baze	Kompesija &baze
Kraj rada	Kraj rada	&Kraj rada

Osobina **Name** je naziv tastera čije se ime upotrebljava u programu, dok je **Caption** tekst koji će se ispisivati na tasterima. Znak **&** (tzv. ampersend) ispred određenog karaktera označava prečicu sa tastature za pristup tom tasteru. U našem primeru ispred karaktera "P" na tasteru "&Pregled" stoji znak **&**, što Accessu (ovo je standardizovano na nivou operativnog sistema) govori da se pritiskom na tastaturu, na slovo "P" aktivira odgovarajući taster na formi, kao da smo levim klikom miša kliknuli na njega. Obratimo pažnju da, dok je forma u fokusu, nije neophodno kombinovati hot-key sa tasterom **ALT**. Za pretpostavku, ako se nekom od menija pristupa preko tastera sa **&P** i ako je u isto vreme aktivna forma koja na sebi ima taster sa **&P**, meniju ćemo pristupiti sa **ALT+P**, a tasteru na formi samo pritiskom na **P**.

Valja obratiti pažnju da se na formi ne postavi isti hot-key za pristup dvoma komandnim tasterima: pri aktiviranju "vrućeg tastera" aktiviraće se prvi na formi.

NAPOMENA
Da bi datum u naslovu komandne table bio formatiran tačno kao na slici, potrebno je da u apletu Regional Settings u Control Panelu, na listu Date, u polje Long Date Style postavite masku datuma dddd, d. MMMM yyyy.

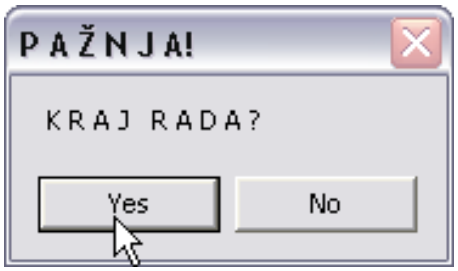
Vratimo se u režim izmene dizajna za ovu formu i pođimo od jednostavnijih događaja. Prvo, za taster "Kraj" postavimo sledeći program u njegov događaj **OnClick**:



```
Dim Response As Integer
Response = MsgBox("Kraj rada?", vbYesNo, "Pažnja!")

If Response = vbYes Then
   DoCmd.Quit
Else
   Cancel = True
   Response = acDataErrContinue
End If
```

Klikom na taster "Kraj rada" ili pritiskom na taster (sa tastature!) **k** Access postavlja kontrolno pitanje:



Slika 67: Zatvara Access

Odgovorom na **Yes** (ili samo **y** sa tastature) program završava rad. Odgovorom na**No** (ili **n**) poništava se akcija, preskače se poruka o grešci koju Access generiše i program nastavlja rad. Kontrolno pitanje nije neophodno, ali sprečava slučajan izlazak iz programa, ako operater nehotice pritisne taster **k** na komandnoj tabli.

Forma za pristup unos podataka i izveštaji ćemo pristupati preko tastera "Unos podataka" i "Izveštaji". Za taster "Unos podataka" postavimo sledeći program u događaj **OnClick**:

```
DoCmd.Minimize
DoCmd.OpenForm "frmUnosNovihPodataka", acNormal
```

Ova dva reda programa prvo minimizuju Komandnu tablu i prikazuju formu **frmUnosNovihPodataka**. Nakon što operater zatvori formu **frmUnosNovihPodataka**, Komandna tabla treba da vrati prvobitni oblik. Za taster "Unos podataka" postavimo još jedan programski red u događaju **OnGotFocus**:

```
DoCmd.Restore
```

Nakon što snimimo izmene i kliknemo na taster "Unos podataka", Komandna tabla se minimizuje i otvara se forma za rad unos podataka. Ali, kada na formi za rad za unos podataka kliknemo na taster "Zatvori masku", aktivna forma se zatvara, a Komandna tabla ostaje umanjena! Naime, nama je aktivan i prozor baze i on prvi dobija fokus nakon zatvaranja forme **frmUnosNovihPodataka**. U normalnim okolnostima, prozor baze se drži sakriven i Komandna tabla će prva dobiti fokus, odnosno, aktiviraće se događaj **OnGotFocus** na tasteru "Unos podataka" koji će vratiti formi prvobitnu veličinu.



Proceduru opisanu za taster "Unos podataka" sprovedimo i na njemu susedni - "Izveštaji" sa jednom izmenom: umesto **frmUnosNovihPodataka** u događaju **OnClick** postavićemo **frmIzvestaji.** Takođe za taster "Pregled" **frmPregled.**

Ovim smo razrešili ponašanje četiri tastera na komandnoj tabli. Na isti način će mo povezati i preostale taster sa odgovarajućim formama.

NAPOMENA:

U programima sa mnogo formi ne praktikuje se stalno otvaranje i zatvaranje formi, jer se tako gubi na brzini rada aplikacije. Umesto metoda **DoCmd.OpenForm** i **DoCmd.Close** koriste se metode **Hide** i **Show**. Naime, sve forme se po startu aplikacije učitaju u memoriji i sakriju se od prikaza. Zatim se, po potrebi, promenom parametara Hide i Show menja njihova vidljivost. Ova operacija je praktično trenutna, ali po cenu povećane potrošnje radne memorije.

Navešćemo tastere, odnosno kombinacije tastera, koji će poslužiti u manipulaciji podacima sa forme:

Tabela 16: Kombinacija tastera za manipulaciji podacima sa forme

taster	značenje
TAB	sledeće polje
SHIFT+TAB	prethodno polje
END	poslednje polje sloga
HOME	prvo polje sloga
CTRL+END	poslednje polje poslednjeg sloga
CTRL+HOME	prvo polje prvog sloga
F2	editovanje sadržaja polja
ENTER	sledeće polje
ESC	poništavanje unosa sloga
CTRL+F4	zatvara formu

U modu editovanja polja

Tabala 17: Značenje tastera u modu editovanja polja

taster	značenje
ENTER	Kraj editovanja, skok na naredno polje
ESC	Poništavanje unosa u polje



21.1 Potvrda upisa sloga kod unosa na formi

Neke osobine okruženja Accessa nisu baš prikladne za svaku priliku. Šta ćemo učiniti da sprečimo nehotičnu promenu sloga?

U Accessu se podaci zapisuju u polje i ceo slog u tabelu, bez obzira na to da li je korisnik prešao na naredni ili prethodni slog ili je zatvorio tabelu. Isto pravilo važi i ako se procedura za unos realizuje preko forme. Praktično, dovoljno je započeti novi unos i slog je zapisan u tabelu, osim ako se tokom unosa ne pritisne dva puta taster Escape (prvi put za odustajanje upisa u polje i drugi put za odustajanje upisa sloga u tabelu).

Situacije kada se slog ne sme zapisati u tabelu bez izričite potvrde su česte - naročito kada se moraju popuniti sva polja sloga, odnosno kada se pre završenog unosa sloga forma zatvori. Jedno od mogućih rešenja primenjenih na formu je postavljanje osobine **Cycle=Current Record** (onemogućavanje prelaska na drugi slog) i postavljanje kontrolnog tastera na nju. Funkcija tastera je upis unetih podataka u tabelu i prelazak na naredni slog. Drugo od mogućih rešenja je kontrola unosa za svako polja zasebno i izdavanje upozorenja ukoliko je neko od polja ostalo prazno.

Varijante izbegavanja automatskog ažuriranja sloga su različite, a ova koju ovde prezentujemo je univerzalna, bez obzira na broj polja. Primenjeno rešenje postavlja kontrolno pitanje pre zapisa sloga u tabelu, odnosno pre zatvaranja forme sa započetim unosom podataka i u zavisnosti od odgovora upisuje slog ili uz pomoć kombinacija Cancel/Undo odustaje od upisa.

U događaj BeforeUpdate forme unesite sledeći program:

```
Private Sub Form_BeforeUpdate(Cancel As Integer)

' Pita da li da upiše slog ili da se odustane
' od upisa prilikom zatvaranja forme sa zapocetim
' unosom ili prelaskom na prethodni/naredni slog

If MsgBox("Save?", vbYesNo) = vbNo Then
Cancel = True
Me.Undo
End If
End Sub
```



21.2 Aktiviranje prekidača (ChekBox) na formi

Prilikom dizajniranja forme često se ukazuje potreba za postavljanjem najmanje jednog prekidača, a možda i više njih. Evo predloga kako da inicijalizujemo prekidače pri otvaranju forme.

U situacijama kada više prekidača čine izbor, kada su stanja prekidača međusobno zavisna i njihove vrednosti uključive, a korisnik pri tome želi sve sem jednog da isključi (uključi), postavlja se komandni taster za menjanje stanja svih u **False (True).**

U ovakvoj situaciji manji broj prekidača ne predstavlja veliku muku za isključivanje putem VBA programa. Jednostavno se taksativno pobroje svi prekidači i dodeli im se vrednost **False**:

```
Private Sub Svi_Click()
' U događaj OnClick tastera
' "Deselektuj sve":

Check1 = False
Check2 = False
Check3 = False
```

odnosno vrednost **True** za **OnClick** događaj na tasteru "Selektuj sve".

Kod velikog broja prekidača na formi, prilično je nezgodno na ovaj način menjati njihove vrednosti, te se pribegava upotrebi svojstva TypeOf. Sledeći način rešava efikasno problem za neograničen broj prekidača, proveravajući tip kontrole:

- Na formu sa prekidačima postavite dva tastera sa nazivima "Svi" i "Nijedan"
- U **OnClick** događaj tastera "Svi" upišite sledeći kod:

```
 Private Sub Svi_Click()

' Selektuje sve prekidače na formi

Dim ctr As Control
For Each ctr In Controls
  If TypeOf ctr Is CheckBox Then
    ctr.Value = True
  End If
Next ctr
End Sub
```



- U OnClick događaju tastera "Nijedan" upišite sledeći kod:

```
Private Sub Nijedan_Click()

' Deselektuje sve prekidače na formi

Dim ctr As Control
For Each ctr In Controls
  If TypeOf ctr Is CheckBox Then
    ctr.Value = False
  End If
Next ctr
End Sub
```

Opisana tehnika je jednako primenjiva i na prekidače tipa "Toggle Button" i "Option Button", a rešenje je univerzalno, pa se bez obzira na naziv prekidača (koji može imati vrednosti **True** i **False**) može primeniti jednostavnim kopiranjem u OnClick događaj odgovarajućeg kontrolnog tastera na formi.

U obrnutom slučaju, kada treba dozvoliti da samo jedan od više prekidača ima vrednost **True**, odnosno kada su stanja prekidača isključiva, može se programski na događaj **OnClick** svakog prekidača postaviti kôd koji će stanja ostalih postaviti na vrednost **False**. Problem se javlja kada ovu operaciju treba sprovesti na velikom broju prekidača.

Rešenje ovog problema ne leži u programiranju, nego u sledećem postupku:

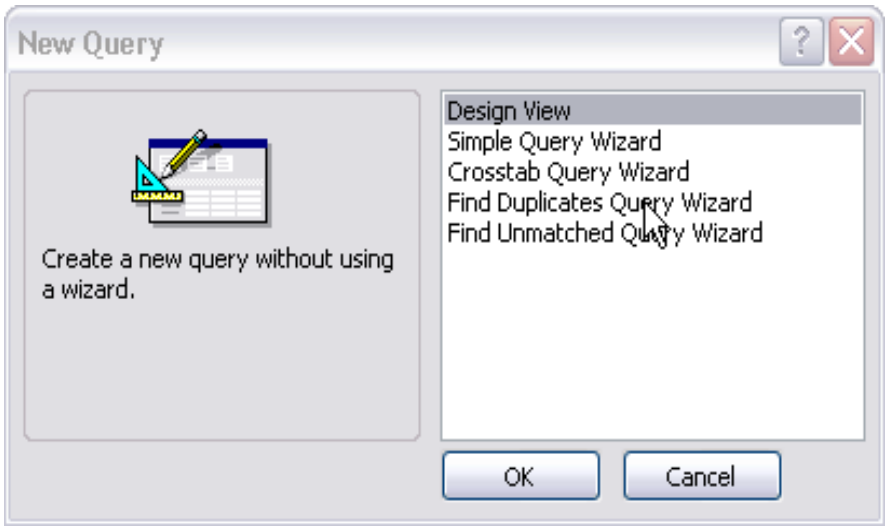
- Otvorite formu u dizajn-pogledu.
- Ako na ekranu nemate grupu ikona "Toolbox", uključite je.
- Odaberite ikonu "Option Group" i postavite je na formu.
- Odaberite ikonu "Check Box" i postavite je unutar "Option Group".
- Ponavljajte tačku 4 dok na formu ne postavite potreban broj prekidača.

Otvorimo formu i promenimo vrednost jednog prekidača u vrednost True (uključen). Ostali unutar te pseudo-grupe će dobiti vrednost False (isključen). Ovim primerom se ilustruje rešavanje problema bez programiranja, što uvaženo mišljenje da se u Accessu skoro ništa ne može uraditi bez korišćenje VBA samo potvrđuje kao izuzetak.



22 QUERY

Pređimo na karticu upita i pritisnimo taster **New**. Iz ponuđenog menija biramo **Design View** i pritiskamo taster **OK**.



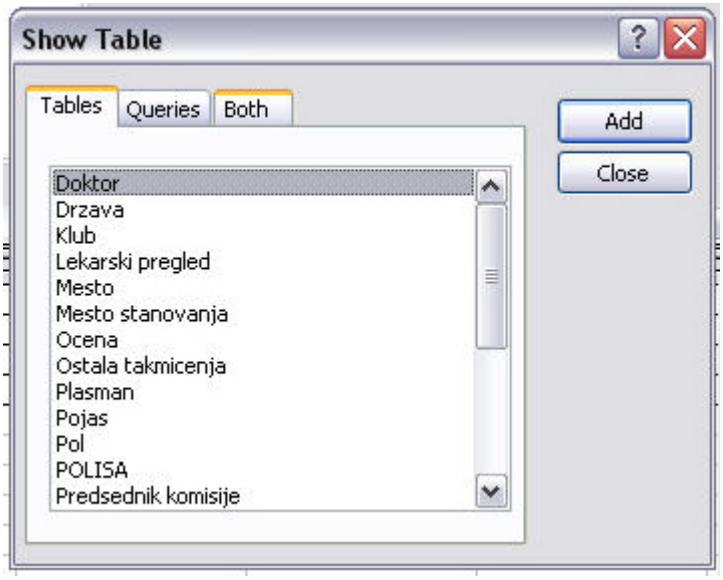
Slika 69: Izbor opcija za novi Query

Zadržimo se na moment na ovom meniju. Primetimo da samo **Design View** nema sufiks **Wizard**. Access je, kako smo već pomenuli, prepun ovih "čarobnjaka", što su, u stvari, poluautomatizovane procedure za postizanje određenog cilja uz minimalno angažovanje korisnika. Ovaj put ćemo samo pomenuti namenu ostalih opcija, jer kada upite savladamo kreirajući ih samostalno od početka, shvatanje ostalih opcija je jednostavno.

Značenje opcija:

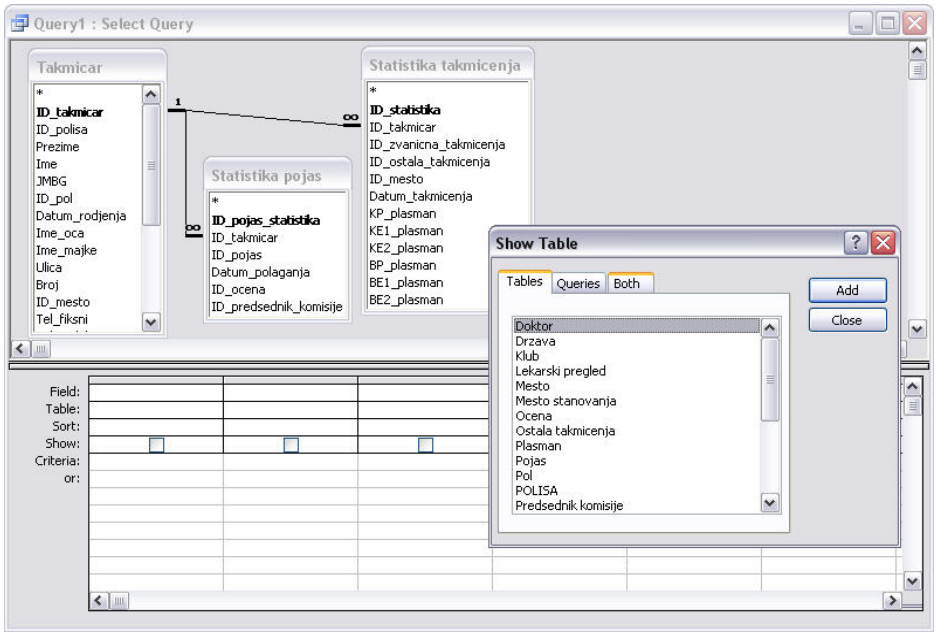
- **Simple Query Wizard** je procedura koja pomaže u sastavljanju jednostavnih upita. Sastavljanje ovakvih upita je brže kada se radi bez upotrebe "čarobnjaka";
- **Crosstab Query Wizard** predstavlja proceduru za sastavljanje upita za unakrsne proračune. Ukratko, unakrsni proračuni predstavljaju grupisanje rezultata po više kriterijuma (npr. za jednog takmičara i u određenom vremenskom intervalu tražimo sumiran rezultat sa takmičenja);
- **Find Duplicates Query Wizard** sastavlja upit koji prikazuje samo one podatke koji se ponavljaju u dve relaciono povezane tabele;
- **Find Unmatched Query Wizard** sastavlja upit koji prikazuje samo one podatke jedne tabele koja se ne pojavljuje u drugoj, relaciono povezanoj tabeli.

Posle klika na taster **OK** dobijamo spisak svih tabela i upita naše baze. Upit može podatke preuzimati direktno iz tabela ili iz drugih upita.



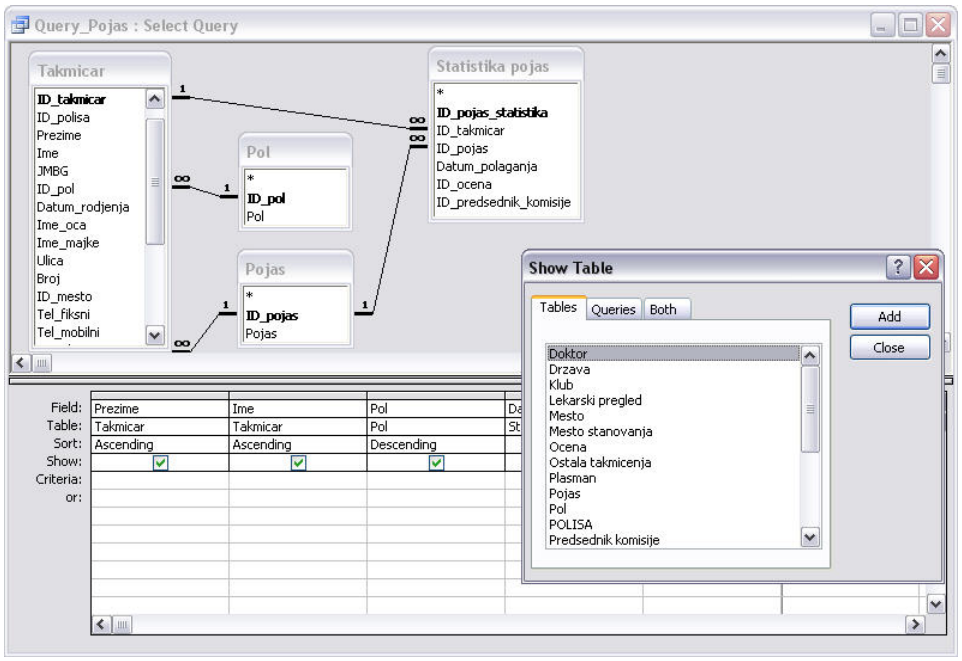
Slika 70: Show table opcija za dodavanje tabela u prozoru Query

Kliknimo na tabelu **Takmicari** i na taster **Add**. Zatim kliknimo na tabelu **PojasStatistika** i opet na **Add** i konačno ponovimo postupak za tabelu **Pojas**. Zatvorimo ovaj meni pritiskom na taster **Close**. Na ekranu bi trebalo da dobijemo upit nalik slici:



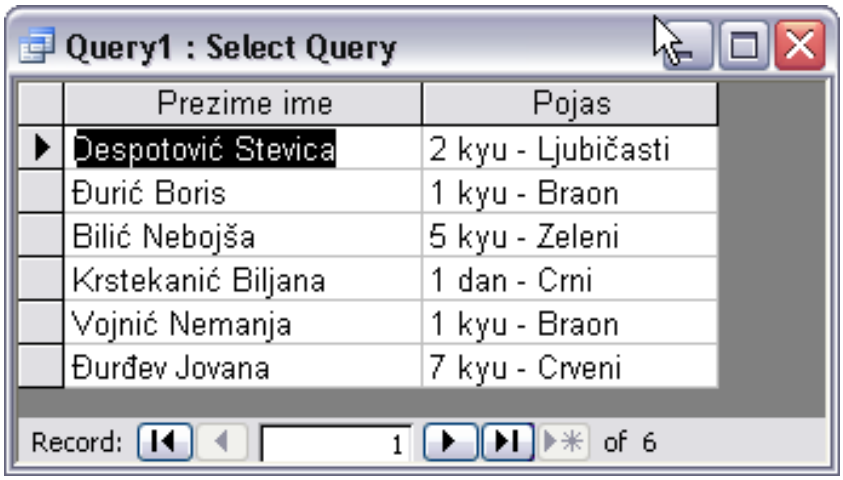
Slika 71: Izgled Query prozora

Ovaj upit za sada ne radi ništa, jer nismo naveli iz koje tabele će se koji podaci izdvajati za prikaz. Tabele koje se postavljaju u upit **moraju** biti povezane preko određenih podataka koji će biti isti za tabele u relaciji. Pogledajmo naš upit: iz tabele **Takmicari** kolona **ID_takmicar** je u direktnoj relaciji sa istoimenom kolonom tabele **PojasStatistika** i podaci koji će se pojavljivati u ove dve kolone su direktno zavisni. Praktično, iz ove tri tabele možemo izdvojiti bilo koju kolonu iz bilo koje tabele, dok god su one preko nekog ključa direktno povezane, a to je u ovom slučaju **ID_takmicar**. Pogledajmo kako to radi:



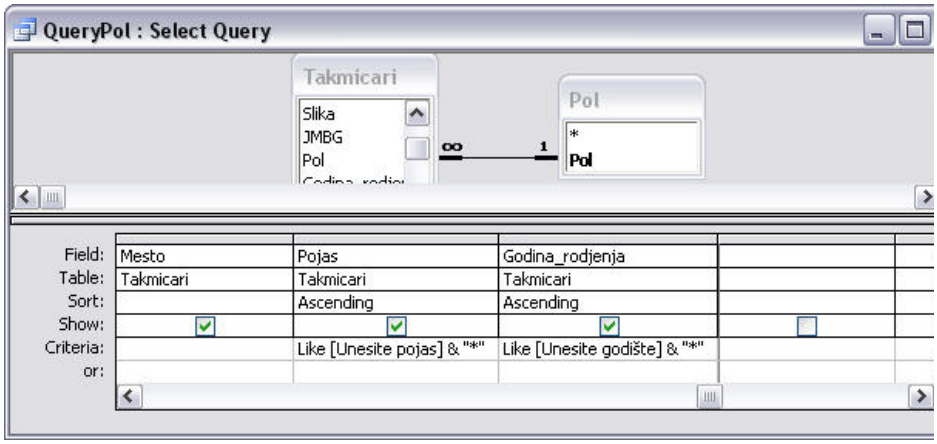
Slika 72: Query pojas

Kliknimo na taster za izvršavanje upita i pogledajmo rezultat:



Slika 73: Tabelarni prikaz Query pojas

Upit je izdvojio članove (prikazuje samo ime i prezime, kako smo i zahtevali) i naziv pojasa sa kojom je takmičar u vezi. Kliknimo sada na taster za prelazak u **Design View** i vratimo se na dizajniranje upita.

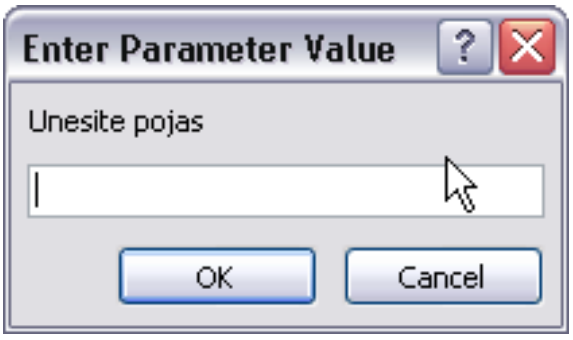


Slika 73: Query pol sa kriterijumom

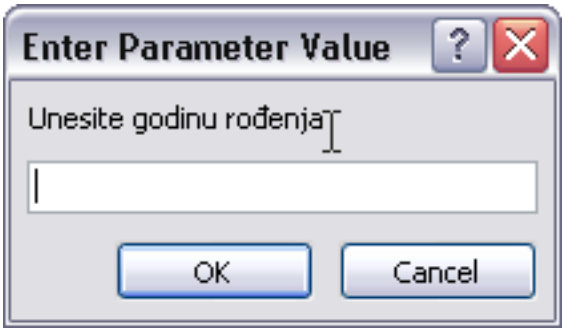
U koloni **Prezime_ime** smo za način sortiranja postavili prikaz na rastući redosled, odnosno po ovoj koloni će biti sortirani rezultati. Na upit smo postavili tri kriterijuma po kojima će podaci biti izdvojeni.

Like [Unesite pojas]&"*"
Like [Unesite godinu rođenja]&"*"
Like [Unesite pol]&"*"

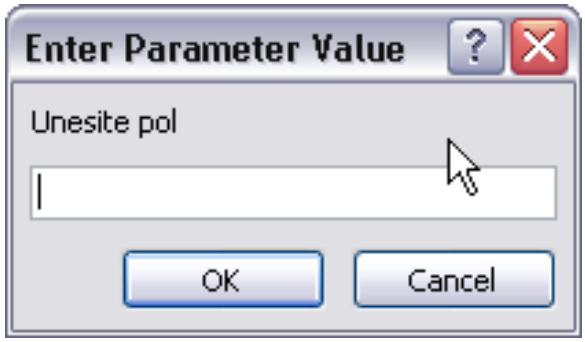
Pokrenimo upit. Pre prikazivanja rezultata upitani smo da odredimo pojas, godinu rođenja i pol. Ako na postavljeno pitanje samo odgovorimo sa OK bez kriterijuma, upit će uzeti u obzir celu kolonu sa svim podacima.



Slika 74: Možemo da unesemo podatak o pojasu



Slika 75: *Možemo da unesemo podatak o godini rođenja*



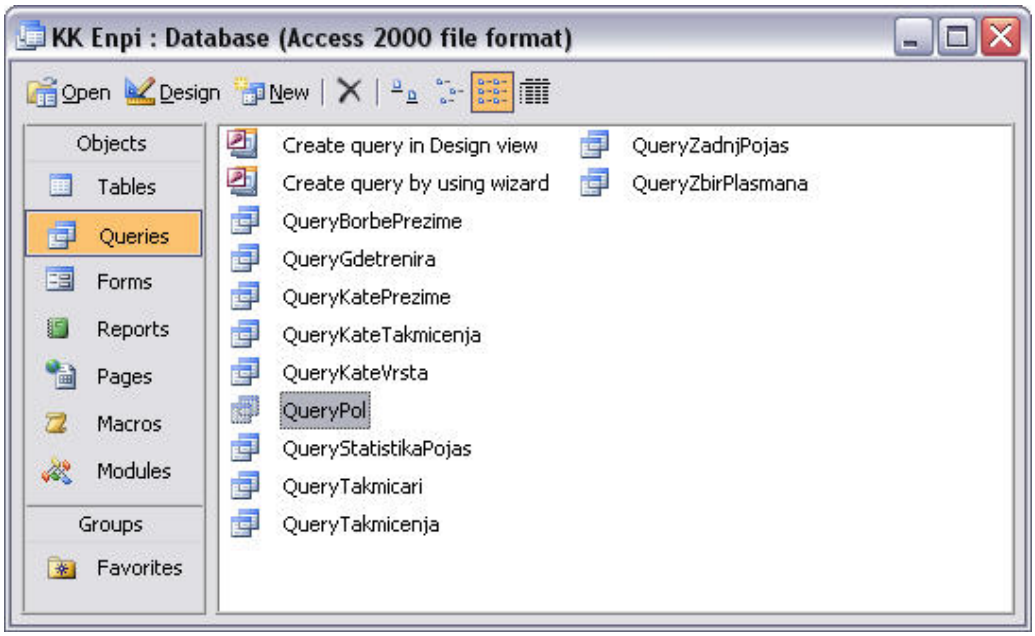
Slika 76: Možemo da unesemo podatak o polu

Rezultat upita za selektovani 4 kyu – Plavi prvi pojas, bez godišta i pola ima oblik tabele kao na slici 77:



Slika 77: Izgled Querya

Na osnovu ovog i sličnih upita kreiraćemo razne izveštaj pod odgovarajućim nazivom.

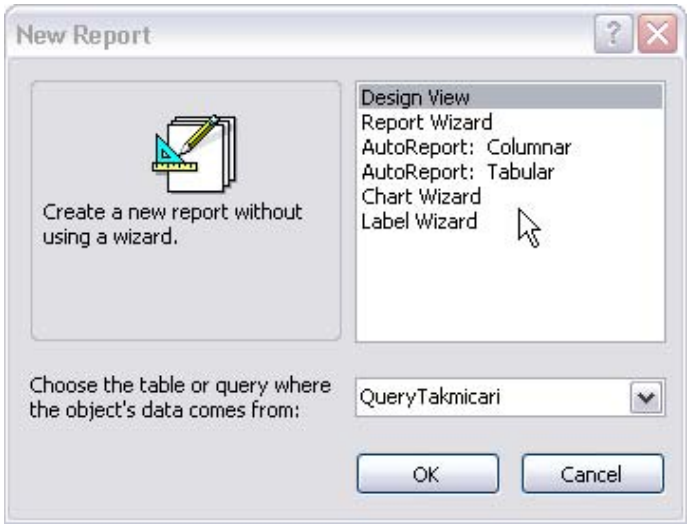


Slika 77a: Izgled Querya prozora

23 IZVEŠTAJI

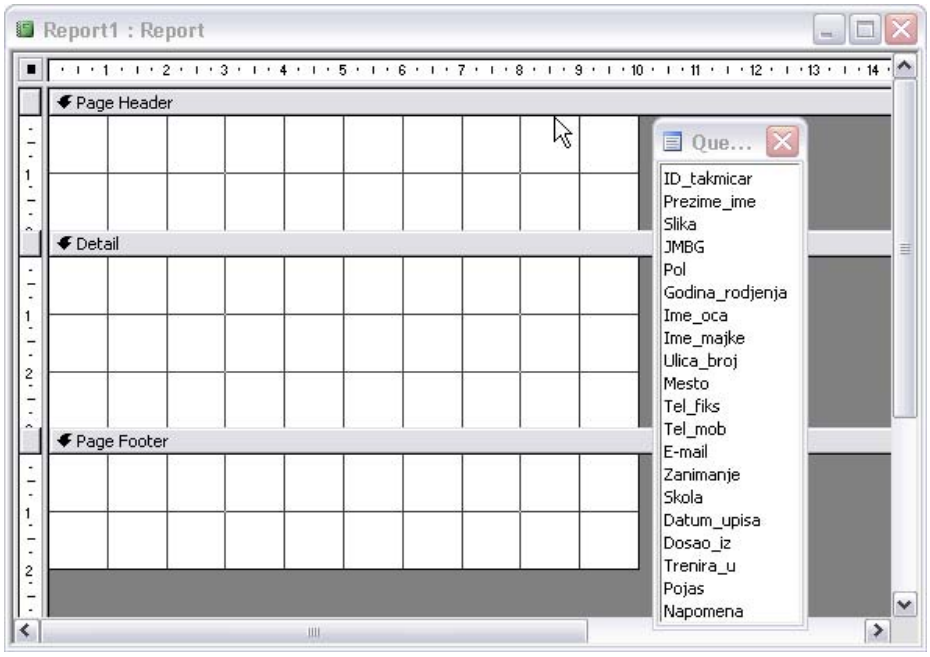
Prvi izveštaj ćemo kreirati za taster Komandne table **frmIzvestaji** "član kluba". Izveštaj je jednostavan i ne zahteva posebne kriterijume upita, jer prikazuje sve članove u našem klubu.

Pređimo na karticu **Reports**. Klikom na taster **New** i iz menija odaberimo **Design View**, iz padajućeg menija izaberimo QueryTakmicari. Kliknimo na OK, spisak polja table takmičari dobijamo u zasebnom prozoru.



Slika 78: Izbor opcija za kreiranje novog izveštaja

Pre nego što pređemo na dizajniranje izveštaja i raspoređivanje polja, objasnićemo izgled "sirovog", praznog izveštaja.



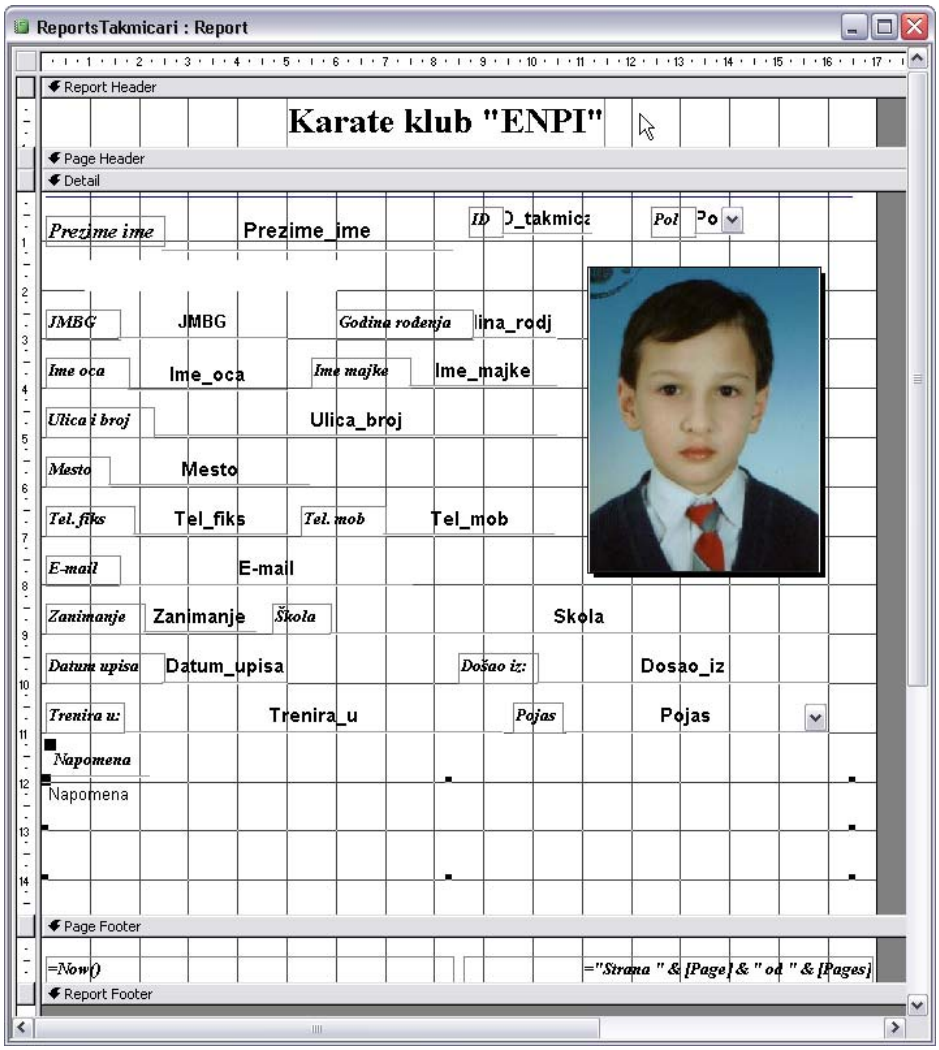
Slika 79: Izgled Design View prozora za izveštaj QueryTakmicari

Osnovni izveštaj je podeljen na tri sekcije. U prvu, gornju, pod nazivom **Page Header** se postavlja naslov izveštaja, eventualno datum i neki drugi sličan sadržaj koji se pojavljuje **samo** na prvo strani. U drugu sekciju pod nazivom **Details** se postavljaju polja koji su nosioci podataka. Tu se može postaviti i tekst, ali će se on za svaki zapis iz tabele ili upita ponavljati za svaki prikazani slog. U poslednju sekciju **Page Footer** postavlja se sadržaj koji se nalazi **samo** na kraju izveštaja.

Izveštaj treba da je pregledan i jednostavan, uz minimalnu količinu dodatnih informacija.

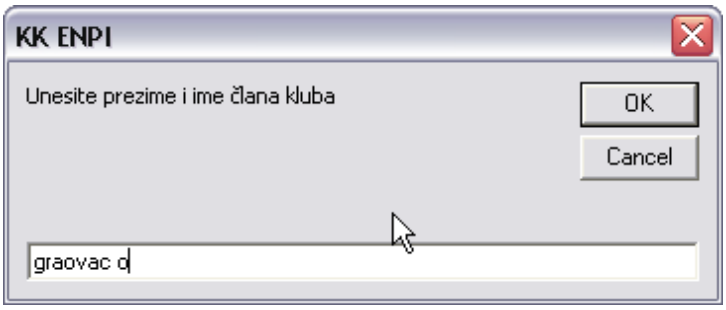
U sekciju **Report Header** postavićemo jednu labelu u kojoj ćemo upisati **Karate klub „ENPI“**. Uravnajmo centralno u odnosu na list ovu labelu.

Iz liste polja tabele **QueryTakmicari** selektujmo prvu stavku **ID_takmicar** i prevucimo je na izveštaj u drugu sekciju – **Detail.** Rasporedimo i ostale stavke prema spostvenom izboru.



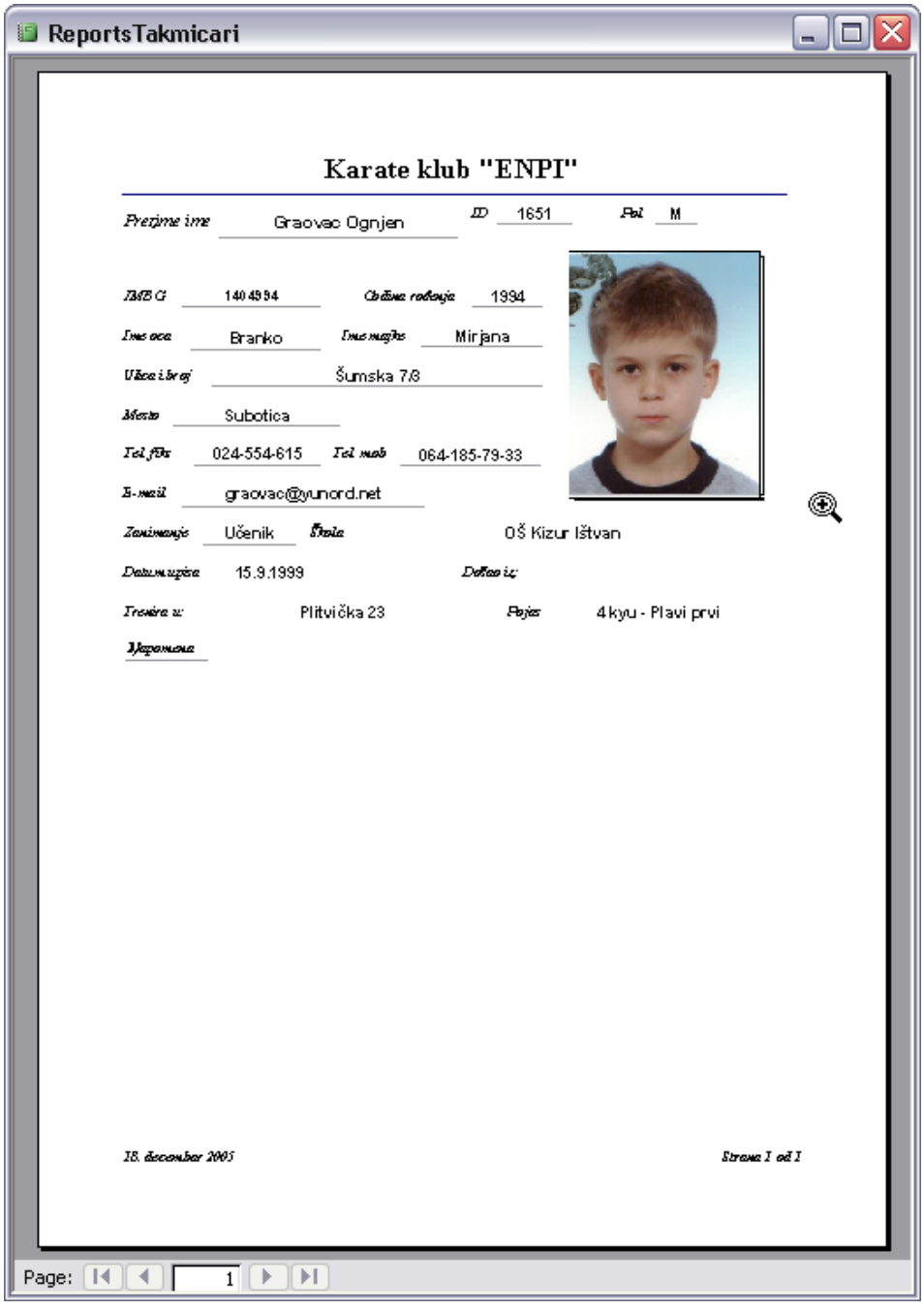
Slika 80: Raspored polja za prikaz izveštaja

Konačno, snimimo izveštaj pod imenom **ReportsTakmicari** i proverimo njegov izgled pritiskom na taster **Preview**.



Slika 81: Unesimo prezime člana za koji želimo izveštaj

Izveštaj smo radili na osnovu upita **QueryTakmicari** gde smo postavili kriterijuma po kojima će podaci biti izdvojeni. Ako ne unesemo prezime i ime i kliknemo na taster OK, dobićemo izveštaje za sve članove kluba. Ako na postavljeno pitanje odgovorimo kao na slici, dobijamo sledeći izveštaj:



Slika 82: Izgled traženog izveštaja spremnog za štampu



Vratimo se Komandnoj tabli tasteru član kluba i događaju **On Click** pridružimo sledeći programski red:

```
DoCmd.OpenReport "ReportsTakmicari", acViewPreview
DoCmd.Maximize
```

U događaj **On Got Focus** postavimo:

```
DoCmd.Restore
```

Ako u programskom redu izostavite **acViewPreview** klik na taster će štampati izveštaj, umesto da ga prikaže na ekran.

23.1 Sprečimo pojavu praznog izveštaja

Zašto da gubimo vreme na izradu izveštaja ako nema podataka koji su rezultat upita?
Postoji vrlo jednostavan način da naložimo Accessu da ne formira izveštaj ako nema nijednog sloga koji odgovara upitu. Iskoristimo svojstvo NoData objekta Report - u događaj NoData modula izveštaja ubacite sledeći kod:

```
Private Sub Report_NoData (Cancel As Integer)

Msgbox "Ovaj izveštaj nema aktuelnih podataka."
Cancel=True

End Sub
```

Umesto da se otvara novi prozor izveštaja u kome ćemo tek pregledom utvrditi da podataka nema, dobićemo poruku, a procedura će biti prekinuta. Naravno, ovo jezgro možemo upotrebiti da bismo uspostavili i čvršću kontrolu slične situacije.



24 ZAVRŠNE RADNJE UREĐENJA PROGRAMA

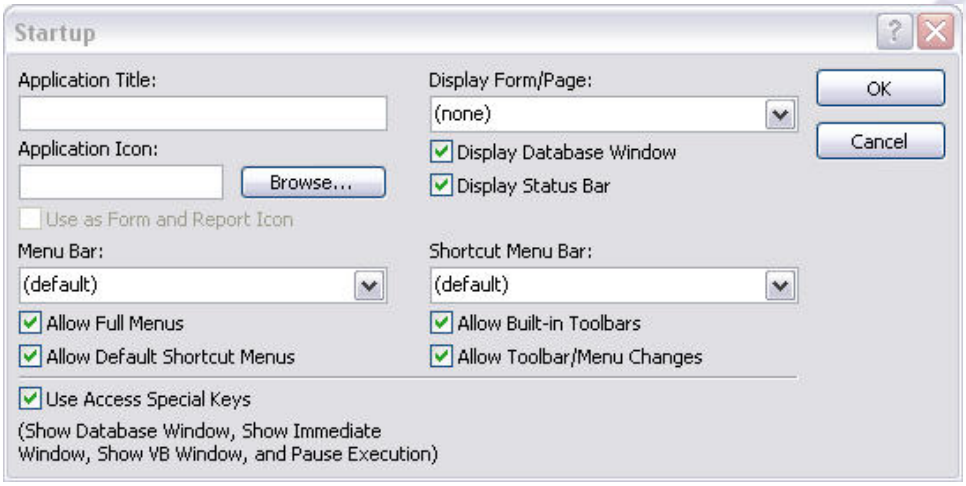
Sada ćemo u našoj aplikaciji postaviti menije koji će zameniti standardne Accessove, prevešćemo bazu u MDE oblik kod koga se ne može menjati program, postavićemo pozdravni ekran i analizirati načine zaštite baze. Predvidećemo da se sa našom aplikacijom može raditi na više radnih stanica i videti na koji način se ovo može najefikasnije učiniti.

Ovo je, ujedno, poslednji stadijum u razvoju aplikacije.

Kada korisnik pokrene našu aplikaciju, odnosno kada (dvo)klikne na odgovarajuću ***.mdb** datoteku, pokreće se Access sa svojim pozdravnim ekranom i učitava bazu. Kroz sledeći postupak podesićemo aplikaciju da se po učitavanju sakrije prozor baze i da se odmah učita prva forma **frmKomandnaTabla**.

Kliknimo na meni **Tools** i zatim na stavku menija **StartUp**....

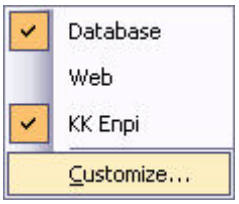
- u polje **Application Title** upišimo **Karate klub**
- za **Application Icon** možemo upisati putanju do ikone kojom bismo zamenili standardnu Accessovu (sasvim gornji levi ugao programa)
- **Menu Bar**: ćemo za sada ostaviti na osnovnoj vrednosti. Kasnije ćemo kreirati meni koji će zameniti postojeći**
- isključićemo **Allow Full Menus** i **Allow Default Shortcut menus**. Ovim onemogućujemo pun sadržaj osnovnih menija i osnovni meni koji se dobija klikom na desni taster miša
- u **Display Form/Page**: ćemo upisati (ili izabrati iz liste) **frmKomandnaTabla**. Ovim naređujemo da se po startovanju baze odmah učita komandna tabla
- isključimo **Display Database Windows**. Sakrivamo prozor baze
- **Display Status Bar** - možemo ga isključiti, ali ne smeta. **Status bar** je donja linija programa sa korisnim informacijama kao što su **NUM LOCK** uključen ili ne, **CAPS LOCK** uključen ili ne, program je spreman za rad **(Ready)** i slično
- isključimo još i **Allow Built-in Toolbars** i **Allow Toolbar/Menu Changes**
- kliknimo na dugme **Advanced** i isključimo **Use Access Special Keys**. Ovim oslobađamo sve **posebne** funkcijske tastere (npr. pritisak na **F11** prikazuje prozor baze; ovim ga isključujemo)



Slika 83: Prozor StartUp

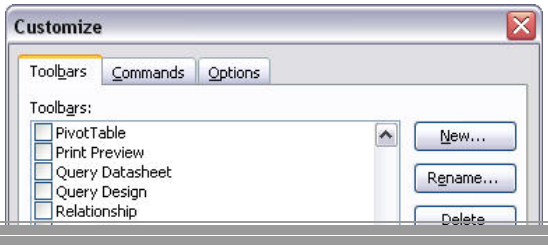
Zatvorimo bazu i ponovo je otvorimo. Prozor baze je sakriven i startovana je komandna tabla. Da bismo nastavili sa izmenom aplikacije, moramo sada zatvoriti kompletan program, jer nam je isključen prozor baze i sve dodatne funkcije za bilo kakvu izmenu aplikacije. Startujemo sada Access ponovo, i kroz dijalog **File/Open** pronađimo našu bazu. Sada **uz PRITISNUT taster SHIFT** kliknimo na **Open**. Držanjem tastera **SHIFT** u toku startovanja aplikacije preskaču se sve opcije koje smo postavili kroz dijalog **Start Up**.

Kliknimo desnim tasterom na bilo koju ikonu toolbara ili bilo koju stavku menija i iz menija izaberimo Customize.



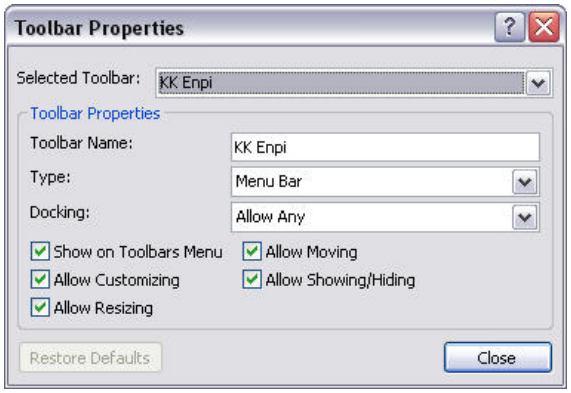
Slika 84: Padajući prozor za izbor opcije Customize

Na kartici **Toolbars** pritisnimo taster **New** i upišimo KK Enpi. Pritisnimo taster za izmenu osobina Properties.



Slika 85: Izgled prozora Customize

Za Type postavimo Menu Bar i uključimo Show On Toolbars Menu.



Slika 86: Izgled prozora Properties

Ovim kreiramo novi meni koji će biti padajućeg tipa, a njemu ćemo moći pristupiti desnim pritiskom tastera na menije.

Zadržimo se još uvek na dijalogu Customize i pritisnimo desnim tasterom na meni KK Enpi. Prevucimo sada levim tasterom miša naš meni neposredno ispod standardnog Accessovog:



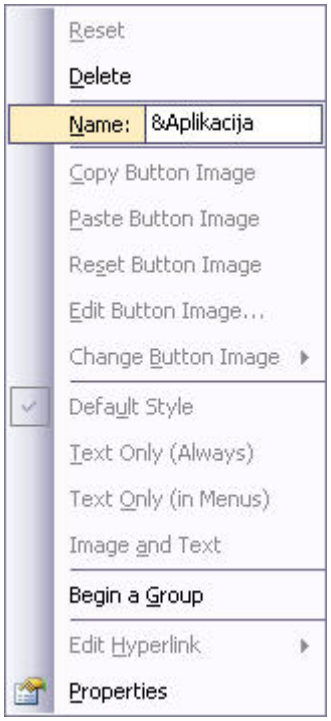
Slika 87: Prikaz praznog menija

Pređimo na karticu Commands i u listi Categories: skrolujmo skroz do poslednje stavke New Menu. Iz desnog prozora Commands: prevucimo levim tasterom miša New Menu na naš Toolbar:



Slika 88: Izaberimo New Menu

Pritisnimo desnim tasterom na New Menu natpis koji se nalazi u Toolbaru KK Enpi i u polje **Name** upišimo &Aplikacija.



Slika 89: Imenovanje novog menija

Sada iz prozora Customize prozora odskrolujmo gore do menija File u listi Categories, kliknimo na File i iz spiska komandi pronađimo Exit i prevucimo ga u meni Aplikacija.



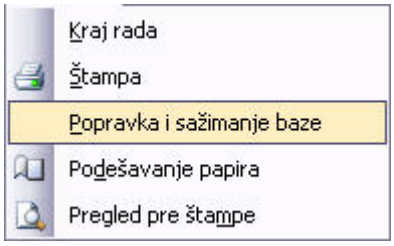
Slika 90: Komanda Exit u meniju Aplikacija

Izmenimo natpis Exit u &Kraj rada. Ponovimo sada postupak sa prevlačenjem komande **Print** iz menija **File** (menjamo natpis u &Štampa); za komandu Compact And Repair Database iz menija Tools (menjamo natpis u &Popravka i sažimanje baze) i konačno za komandu frmPoslovanje iz menija All Forms.

Zatvorimo dijalog Customize. Nakon ove "slikovnice" - pravolinijskog postupka kojom se kreira meni, u Accessovom meniju će se naći i naš meni:

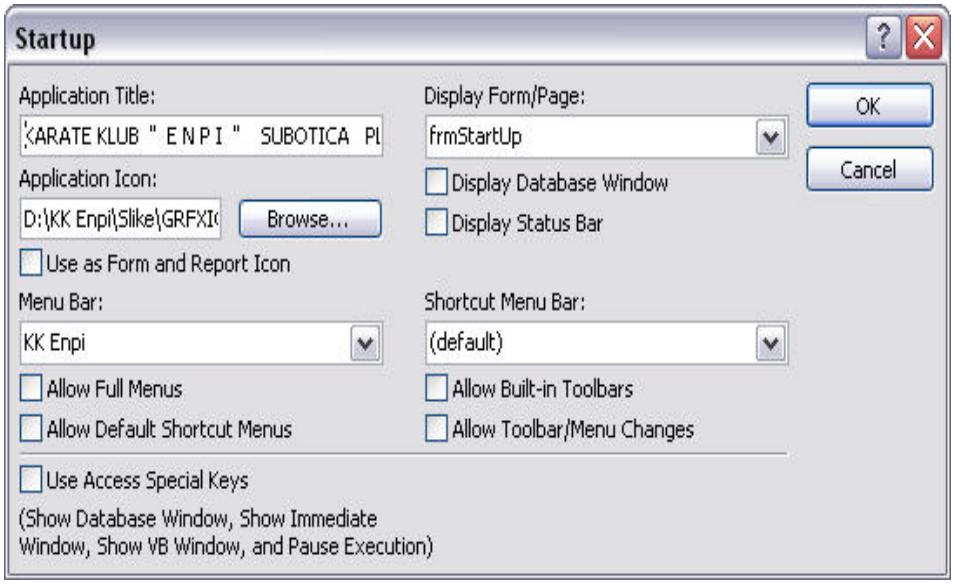


Slika 91:
Izgled našeg menija



Vratimo se sada na Tools i Start Up. Sada iz liste Menu Bar možemo odabrati meni KK Enpi. Podesimo i ostale opcije kao na slici 92.

Slika 92:
konačan izgled
opcije Startup



Ovim smo definisali kompletan program, uključujući menije i izgled ekrana.

25 POZDRAVNI EKRAN

Pozdravni ekran postoji gotovo u svakoj "ozbiljnoj" aplikaciji, naročito u onoj koja se relativno duže učitava, čime daje poruku da je proces učitavanja u toku. Sam pozdravni ekran se može u Accessu dobiti na dva načina. Za prvi način, koristimo se malim trikom, dok je drugi programski iz same baze.

Drugi način je da se koristi zasebna forma koja se sama zatvara. Potrebna nam je prazna forma u koju ćemo postaviti odgovarajući tekst i sliku. Napravićemo formu **frmStartUp** koja se sama zatvara nakon tri sekunde i pokreće formu **frmKomandnatabla**. Pogledajte u režimu izmene dizajna forme **frmStartUp** stavke **On Timer** i **Timer Interval**. **Timer Interval** vreme je dato u milisekundama, a nakon prolaska definisanog vremena pokreće se program naveden pod **On Timer**. Obratite pažnju da ako koristimo drugi način prikazivanja pozdravnog ekrana, da kroz dijaloge **Tools/Start Up** postavimo pozdravnu formu za prvu koja se automatski pokreće po startovanju baze.



Slika 93: Pozdravni ekran



26 KAKO NAPRAVITI ACCESS APLIKACIJU

Večita rasprava o Accessu: može li se napraviti nezavisni program koji će biti distribuiran kao korisnička aplikacija?

Mnogo puta sam postavljao postavljeno pitanje "kako da napravim EXE verziju aplikacije u Accessu". Pitanje se okreće oko nepotpune informacije koju neki korisnici imaju, a po kojoj postoji način da se aplikacija baze podataka isporučuje drugim korisnicima.

Odgovor glasi:

Ne možete napraviti EXE verziju od MDB datoteke, niti je tako nešto ikada bilo moguće.

Međutim, daleko od toga da rešenje ne postoji: odavno postoji sistem koji dozvoljava da se Access aplikacija spakuje i instalira na računare na kojima nije instaliran Access. U ranijim verzijama Accessa (2.0 i 95) postojao je poseban dodatak po imenu Access Developer's Kit (ADK); paket Office 97 imao je opcioni dodatak Office Developer Tools (ODT), a Office 2000 i noviji postoje u verzijama Developer, koji predstavljaju najobimnija i najkompletnija izdanja paketa. Karakteristični deo Developera je sistem za gradnju integrisanih rešenja, a jedan od glavnih modula je sistem za distribuciju Access aplikacija.

Prvo da razjasnimo: baza podataka ostaje u osnovnom formatu; svakako, kada distribuiramo aplikaciju, MDB datoteku valja da pretvorimo u format MDE, u kome nije moguće "petljati" po strukturi tabela, upita i drugih elemenata baze. Dodatni nivoi zaštite se sprovode kroz modul za administraciju pristupa u programu (Tools/Security). Kada su izvedeni svi potrebni nivoi zaštite, aplikacija se posebnim programom (InstallShield) pakuje na CD, diskete ili na drugi nosilac; pritom se, osim baze, pakuju sve datoteke potrebne za rad aplikacije, što uključuje modul Access Runtime (koji je, za razliku od celog programa, slobodan za distribuciju). Ako je sve izvedeno kako valja, aplikacija se instalira na drugi računar u poznatom maniru instalacije softvera, a u aplikaciju se ulazi duplim klikom na MDE datoteku, koja će pri instalaciji dobiti asocijaciju na runtime modul.

Postoji samo jedan način da obezbedimo pravu EXE datoteku koja je izvršna za našu aplikaciju baze podataka - da sami napišemo program u nekom programskom jeziku (Visual Basic, Visual C++, Delphi...), u kome ćemo se sami pobrinuti oko kompletnog interfejsa za pregled i uređenje podataka, dok u priloženoj MDB/MDE datoteci preostaju samo tabele i upiti. Visual Studio takođe sadrži program za pakovanje distribucionog sadržaja gotove aplikacije, što možemo iskoristiti i za distribuciju svoje Access aplikacije.



27 PODELIMO MDB DATOTEKU NA PODATKE I APLIKACIJU

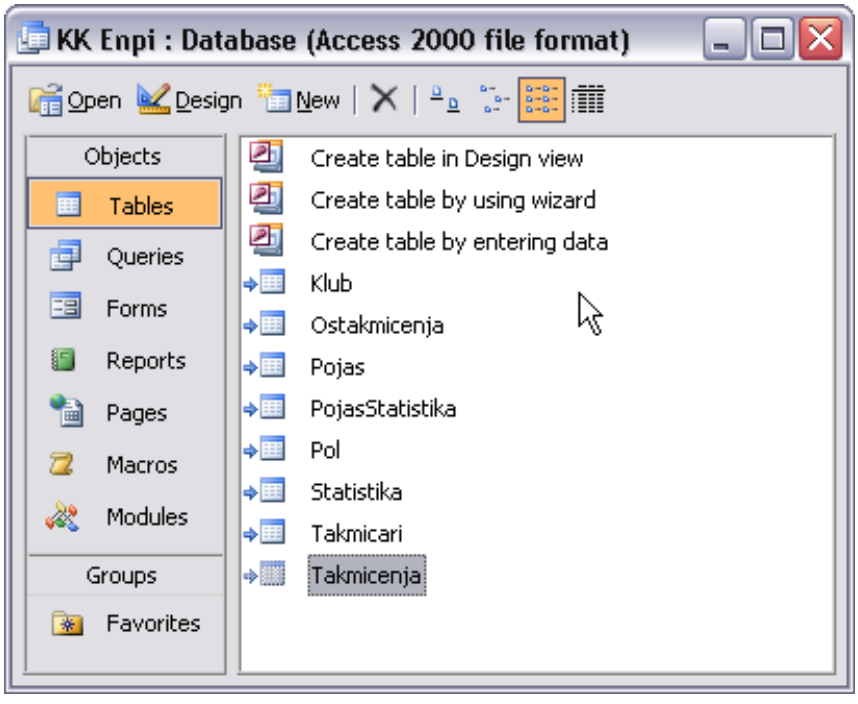
Postoji bar nekoliko valjanih razloga zašto biste MDB datoteku rasparčali na više njih.

Ako intenzivno koristimo aplikacije pisane u Accessu u mrežnom okruženju, potrebno je da razmišljamo o performansama. Često se događa da veći broj korisnika koji istovremeno pristupaju podacima, upitima i obrascima izazove veliko opterećenje u radu aplikacije i zagušenje u saobraćaju mreže.

Pokazuje se da se mogu izvesti koraci koji će dramatično uticati na brzinu izvršenja aplikacije. Podelite datoteku na dve: u prvu datoteku smestite sve podatke, tj. tabele, a u drugu sve ostale elemente aplikacije (upite, obrasce, izveštaje, makroe i module), koji će raditi sa pridruženim tabelama iz prve datoteke.

Iskopirajmo datoteku "KK Enpi.mdb" u "KK Enpi.mdb" i "KK Enpi_be.mdb". Zatim otvorimo u Accessu "KK Enpi_be.mdb" i iz nje obrišimo sve osim tabela. Otvorimo "KK Enpi.mdb" i iz ove baze obrišimo sve tabele. Aplikacija je sada podeljena na dva dela!

Dok smo u bazi **KK Enpi.mdb** kroz meni **File/Get External Data.../Link Tables** pronađimo bazu na disku **KK Enpi_be.mdb** i iz nje povežimo tabele sa otvorenom bazom. Pri svakoj promeni lokacije baze sa tabelama, postupak vezivanja tabela moramo ponoviti!



Slika 94: pogled na linkovane tabele

Drugi dobitak leži u održavanju aplikacije. Iako je potrebno svakom korisniku posebno isporučiti novu verziju datoteke, to je mnogo manji posao nego da se vrši prenos podataka iz stare u novu bazu, jer se tako mogu pogubiti mnogi atributi podataka (recimo, uspostavljene relacije između tabela).



28 RAZLOZI ZA ODRŽAVANJE BAZE

Postoje prilike kada se Access baza mora održavati. Međutim, ima li svrhe da to radite samoinicijativno?

Da li treba redovno raditi operaciju Compact?

Pre izvesnog vremena, prisustvovao sam raspravi u kojoj su sagovornici žučno branili suprotne stavove na temu operacije *Compact* u Accessu: dok je jedan kolega tvrdio da je jedina korist u smanjenju veličine datoteke baze, drugi je insistirao da postoji još korisnih razloga za sažimanje baze. Nisam mogao da presudim ko je u pravu, a onda sam sasvim slučajno naišao na jedan komentar na tu temu, pa ga delim sa vama.

Postoje korisni razlozi da se povremeno primenjuje *Compact* baze, a posebno u višekorisničkom radu. Prvi i najpoznatiji razlog je fragmentacija sadržaja praćena "rupama" u prostoru, što vodi slabijoj performansi u radu baze. Takve baze su manje otporne na oštećenja koja mogu nastati u neregularnim situacijama (blokiranje radne stanice, nestanak struje, kvar diska). Zbog toga se svakako preporučuje da se pre operacije Compact Database kao preventivna mera obavezno primeni operacija Repair Database.

U višekorisničkom okruženju, pre sažimanja baze važe i sledeći preduslovi: korisnik koji obavlja operaciju mora imati dodeljena prava "*Modify Design*" ili "*Administer*" nad svim tabelama baze; na disku mora postojati dovoljno prostora i za originalnu i za saženu verziju baze; konačno, ostali korisnici ne bi smeli da pristupaju bazi u toku operacije.

Sažimanje baze u Accessu daje sledeće rezultate:

- reorganizuje stranice tabele tako da počivaju u neprekinutom nizu (stranica je jedinična količina čitanja tabele);
- poništava neiskorišćeni prostor koji nastaje brisanjem podataka i objekata, čime smanjuje veličinu datoteke; osim toga, ako se ne obriše povremeno, ovaj prostor se kasnije koristi za upis novih slogova baze, pri čemu podaci postaju fragmentirani;
- resetuje brojače u AutoNumber poljima tako da prvi budući broj bude za jedan veći od poslednjeg neobrisanog;
- regeneriše statistiku tabele u procesu optimizacije upita. Vremenom ova statistika može postati neažurna kao posledica poništenih transakcija ili neregularnosti u radu, što nužno dovodi do pada performansi upita ili čak neispravnih rezultata;
- označava sve objekte tipa Query da bi se svaki upit ponovo kompilovao u toku prvog izvršenja. Ovo je naročito važno ukoliko je u toku razvoja radikalno promenjena struktura nekog upita.

Dakle, sažimanje baze sadrži više korisnih posledica i može se preporučiti kao operacija periodičnog "održavanja higijene" baze u Accessu. Ne zaboravite: ovaj postupak je radikalan, pa prethodno obavezno napravite rezervnu kopiju datoteke baze.



29 RAD PROGRAMA SA DVE ILI VIŠE RADNIH STANICA MREŽE

Jasno je da se u režimu rada sa više radnih stanica koje konkuretno pristupaju jednim podacima, svim radnim stanicama podaci moraju biti zajednički i jednako dostupni. Tada se baza mora podeliti na dva dela: jedan deo će sadržati same podatke (tabele) i on će svim radnim stanicama biti zajednički. Drugi deo (upiti, izveštaji, forme i programi) se postavljaju nezavisno na svaku stanicu, dok se pri tome zajedničke tabele povezuju sa svakom distribucijom zasebno.

U praksi, baza sa tabelama se postavlja na najbrži i najbolje obezbeđen računar u mreži, koji može i ne mora biti "posvećeni" server. "Distribucija" se zatim postavlja na svaki računar - radnu stanicu i u nju se tabele postavljene na serveru povezuju u program kroz dijalog **File/Get External Data/Link Tables**.

SAVET:

Iskopirajmo datoteku "Karate klub.mdb" u "bserver.mdb" i "brstanica.mdb". Zatim otvorimo u Accessu "bserver.mdb" i iz nje obrišemo sve osim tabela. Otvorimo "brstanica.mdb" i iz ove baze obrišemo sve tabele. Aplikacija je sada podeljena na dva dela!

Dok smo u bazi brstanica.mdb kroz meni File/Get External Data.../Link Tables pronađimo prvu bazu na disku bserver.mdb i iz nje povežimo tabele sa otvorenom bazom. Pri svakoj promeni lokacije baze sa tabelama, postupak vezivanja tabela moramo ponoviti!



30 PRAKTIČNI SAVETI ZA RAD U ACCESSU

Access je, reklo bi se, složen za upotrebu. Zapravo, samo je potrebno poštovati preporuke, a ostalo je stvar iskustva.

Kreiranje Access aplikacije može biti lako, ali i neugledno i korisniku naporno za rad. Pre svega treba dobro postaviti osnovu - podeliti similarne podatke u zasebne tabele i njih povezati. Pri ovome koristi tačno definisana i određena polja (npr. za datum stavite polje **Date/Time** umesto polja '**Text**' odgovarajuće dužine). Čarobnjake možemo iskoristiti tek da Access postavi potrebna polja na formu ili izveštaj, na nama je da posle toga "dovedemo stvari u red".

Podatke postavimo u zasebne tabele. Ukoliko nismo sigurni koliko tabela treba da bude i koji će se podaci nalazi u kojoj, dodelimo podatak "događaju". Recimo: **NabavkaDatum, NabavkaRobaBroj, NabavkaRobaTip, NabavkaKoličina, KupacBroj, KupacNaziv, KupacTelefon, IsporukaBroj, IsporukaKupacBroj, IsporukaDatum, IsporukaRobaBroj, IsporukaKoličina** itd. Na ovaj način je odmah jasno da je potrebno kreirati tri tabele. Prednosti ovakve organizacije su razne: podaci se ne dupliraju, greške u unosu su praktično izbegnute, a sama aplikacija radi mnogo brže.

Kada imamo ovakvu, stabilnu osnovu - dobro definisane tabele i međusobno povezane, stvari brzo dolaze na svoje mesto, bez nepotrebnog gubljenja vremena na kasnije ispravke. Neki od sledećih saveta pomoćiće u tome da naša aplikacija razvijena u Accessu izgleda bolje.

Pri kreiranju upita (query design) gde je potrebno iz jedne tabele staviti sva polja u tabelu upita, dva puta kliknimo na naslov tabele (caption) i sva polja će biti selektovana. Ostaje samo da ih prevučemo dole u tabelu upita. Obratimo pažnju na zvezdicu koja predstavlja sva polja tabele i može se preneti u tabelu upita, ali se nad tim poljem ne mogu vršiti proračuni.

Osobine forme ili izveštaja se mogu menjati kada se klikne na kvadratić u gornjem levom uglu lenjira.

Osobine grupe objekata se mogu menjati odjednom kada se klikne na prvi, pa uz pritisnut Shift kliknemo i na ostale za koje se želi promeniti neka osobina.

Jedna forma (maska) se može u drugu postaviti kao pod-forma jednostavnim prevlačenjem iz liste formi (u dizajn-pogledu).

Da bi se kreirala pseudo grupa da/ne kontrola na formi, postavimo pravougaonik (rectangle) na formi i u njega postavimo da/ne kontrole (checkboxes).

Cycle properties postavimo na No ako ne želimo da korisnik pritiskanjem tastera Tab doda novi zapis u tabelu.



Na dizajnu forme ili izveštaja, kontrolu možemo precizno pomerati uz pritisnut taster Ctrl i strelicu za pravac, dok se veličina kontrole može precizno menjati uz pritisnut taster Shift i strelicu za pravac.

Ako nam Access da poruku "Query too Complex" verovatno je previše polja i proračuna postavljeno u dizajnu upita. Jedno od rešenja ove situacije bi bilo: pre dodavanja poslednje tabele u upit, formirajmo upit sa svim ostalim tabelama koji rezultat upisuje u tabelu (Make Table query). Ovu, novokreiranu, tabelu zajedno sa poslednjom postavimo u novi upit.

Ako treba napraviti izveštaj sličan formi koja već postoji, otvorimo formu u dizajn-pogledu i snimimo je kao (Save As...) izveštaj. Ostaje da uklonimo kontrole koje se ne mogu prikazati u izveštaju.

Ukoliko je forma prevelika da stane na ekran, razmišljajmo da je razdvojimo na više strana, stavljajući svaku stranu na zaseban list (Tab control).

Umesto unosa datuma putem tastature, na formi se može postaviti pravi kalendar na kojem će korisnik birati datum mišem. Kalendar se dodaje na 'Insert' i 'ActiveX Control...'. "Čitanje" datuma sa kalendara se zatim vrši jednom linijom koda:

```
Me![txtBirthday] = Me![ocxCalendar]. Value
```

Aplikaciju sa ActiveX kontrolom testirajmo na što više računara. Nije isključeno da na nekom drugom računaru neće raditi. Manjak dokumentacije za ove kontrole čini da naša efikasna kontrola ne radi nigde sem kod nas. Npr: na postavljenom kalendaru za izbor datuma je zgodno da kalendar po učitavanju aplikacije pokazuje današnji datum. Znamo li kako se to postavlja? Ovo je loše dokumentovano. Kalendar pamti samo vrednost koju smo u dizajn-pogledu eksplicitno zadali i takav ostaje do sledeće promene 'default' datuma.

Access prihvata razne načine unosa datuma: Prvi januar 2007. se može uneti kao: **1/1/07; 01/01/07; 01,01,07; 01.1.07,** odnosno brojke datuma mogu biti odvojeni bilo kojim separatorom. Ukoliko je u dizajnu tabele za polje datum vrednost **Input Mask** postavljena na oblik **00/00/00** onda je dovoljno za isti datum otkucati **010107** bez separatora.

Greške u dizajnu se mogu vratiti na pređašnjo stanje sa 'Undo'. Obratimo pažnju da svaku grešku moramo odmah vratiti u pređašnje stanje, jer Access dozvoljava samo jedan Undo!

Aplikaciju podelite u dva dela: u jednoj bazi (mdb) se nalaze samo tabele, a u drugoj sve ostalo. Na ovaj način bazu sa tabelama (podacima) nikada ne diramo i uvek se nalazi na lokaciji korisnika; podatke ne moramo da importujemo i eksportujemo; privatnost podataka je veća, a mi možemo aplikaciju menjati ili razvijati menjajući sve ostalo: upite, forme, izveštaje, module... Prednost ovakve organizacije je u mrežnom okruženju - baza sa tabelama se nalazi na serveru, a zasebne aplikacije (maske za unos i izveštaji) se nalaze na radnim stanicama.

Koristimo indekse nad poljima koja se pretražuju. Na ovaj način će sama pretraga biti brža.



Pređimo na VBA. Od Accessa 97 se sve može uraditi iz VBA, dok sa prethodnim verzijama se mora baratiti sa makroima pošto su neke funkcije bile dostupne samo iz makroa.

Koristimo što kraće nazive bez razmaka u njima. Korišćenjem razmaka pri kasnijem definisanju operacija moramo koristi navodnike, a to zahteva više kucanja i veću mogućnost grešaka (primer: **" & Ime & ""**...) što često može dovesti do toga da neki upit ne radi. Razmaci i ostali znaci (izuzev "_") se mogu koristiti ali obratite pažnju pri ukucavanju u VBA tih naziva.

Koristimo uglaste zagrade [] i navodnike gde je potrebno. Jednostavnije je ovo izbeći korišnjem slova i brojeva A-Z, a-z, 0-9 i ako je potreban razmak, znak za podvučeno (_) - underscore. Ukoliko dođemo u situaciju da podatke eksportujemo u, recimo, .dbf format jasne će nam biti prednost ovakvog definisanja imena.

Testirajmo aplikaciju i predvidimo sve greške. Neke greške je dobro zanemariti: korisnik pritisne dugme za izveštaj koje je ograničeno datumima i pri unosu datuma se predomisli i pritisne Cancel - Access prijavljuje grešku. Ovu grešku je nepotrebno prijavljivati (**On Error Resume Next**), dok neke greške je zgodno presresti i uputiti poruku na srpskom jeziku (On **Error GoTo Greska**; i kasnije **Greska**:)

Jednostavne, neblješteće boje, u ograničenom broju su efikasnije od sjajnog i bleštećeg izgleda aplikacije. Nikoga nećemo fascinirati ako budemo isprobavali svoje dizajnerske sklonosti usred poslovne aplikacije.

Ukoliko imamo više velikih formi koje se naizmenično otvaraju i zatvaraju, učitajmo sve forme po startu aplikacije i zatim menjanjem parametra **Visible** prikazujmo ih i sakrivajmo. Brzina rada je mnogo veća, a rad sa diskom mnogo manji.

Osobine forme se mogu menjati na dva načina: pr.

```
'me.visible = false'
```

ili

```
'me![forms]![IDKupac].visible = false'
```

U čemu je razlika? Osobina kontrole odvojena tačkama se koristi kada se podrazumeva forma u fokusu, dok se osobina kontrole odvojena znacima uzvika koristi za bilo koju učitanu formu. Na primer, otvaranje izveštaja za pregled sa filterom na neku kontrolu sa forme bi glasila ovako:

```
DoCmd.OpenReport "Izveštaj", acViewPreview, ,
"[zemlja]=Forms![zemlje]![zemlja]"
```



31 ZALJUČAK

Za savladavanje Accessa je potrebno više vremena nego za sve ostale programe Office paketa. Kreiranje jednostavne baze, kakva je ova za poslovanje karate kluba, ne zahteva veliko poznavanje programa, jer većinu poslova mogu pokriti čarobnjaci. Rezultate rada čarobnjaka ćemo izmeniti po potrebi i program je tu. Ostaje da se malo programira, šminkaju forme, prilagođavaju izveštaji...

Međutim, kada Access treba poterati do maksimuma, tada ćemo prethodnu konstataciju. Sem Accessa, treba upoznati funkcionisanje drugih Office programa u sprezi sa Accessom (često se rezultati iz baze šalju Excelu na analizu ili Wordu na publikovanje), API funkcije Windowsa, rad sa SQL Server aplikacijom u većim mrežama, itd. Ako na ovo dodamo i Internet, spisak se proširuje na HTML i XML (XML je uveden tek u verziji 2002), na Data Access Strane i Internet protokole. Sem svega ovoga, valja obratiti pažnju i na ciljnu verziju koju hoćemo da savladamo. U Accessu 97 je postojao DAO. U verziji 2000 on je, uslovno rečeno, zamenjen sa ADO, dok je u verziji 2002 uveden i ADO MD (multi-dimenzionalna klasa objekata). DAO i ADO se prilično razlikuju, kako u filozofiji rada, tako i sintaksi. ADO MD je blizak ADO biblioteci i može se shvatiti kao proširenje ADO-a. A ova aplikacija što sam napravio, tek pomalo dodiruje programiranje u VBA...

Access je u svojim prvim verzijama bio za tadašnje računare toliko zahtevan program, da su samo oni sa veoma skupim računarima mogli udobno da rade. Oni koji su po firmama imali novije računare mogli su samo da se naoružaju strpljenjem i činjenicom da izgubljeno vreme čekajući da računar obradi zadatak, troše u 8 radnih sati, te da se upuste u avanturu zvanu Access. Mi smo u prednosti: akruelni Access na današnjim računarima "leti", memorija nije skupa, procesorske snage ima napretek, a tu je i skup alata koji će dodatno povećati produktivnost i olakšati rad kroz ono što je nazvano "User-friendly" interfejs. Literature ima mnogo više nego ikada. Access je sada inteligentan: pođe li nešto naopako, pokušaće sam da "popravi" grešku.

Tu je i Microsoft koji se od skoro zvanično predstavio našem IT tržištu. "Original" (koji manje-više što je isti kao kod lokalnog pirata) koji se može kupiti kod legalnih zastupnika donosi i literaturu, tehničku podršku i povećanu sigurnost u vidu ispravnosti kopije.

Što se tiče Accessovim konkurenata, čini se kao i da ih i nema. Zahvaljujućim svom prisustvu u Office Professional paketu, Access je imalo prilike da proba i upozna najširi krug korisnika, a kako su baze veoma upotrebljive kako za kućnu tako i za poslovnu primenu, eto recepta za uspeh.



32 LITERATURA

[1] Microsoft Office, Access 2003, Alison Balter
[2] Microsoft Access 2000 Biblija, Cary N. Prague i Michael R. Irwin
[3] Access 2000 Priručnik za programere, Paul Litwin; Ken Getz; Mike Gilbert
[4] Microsoft Access MODUL 5 Baze podataka, Milan Milosavljević; Mladen Veinović; Igor Franc
[5] dr Angelina Njeguš: Mateijali sa predavanja iz PIS, Fakultet za poslovnu informatiku, Subotica, 2006.
[6] dr Alempije Veljović: Modeliranje informacionih sistema
[7] www.personalmag.co.yu
[8] www.interbest.co.yu



Karate klub Enpi – Subotice – Braće Majera 14
Tel./faks: 38 – 135, Mobil: 063 566 346
www.enpi.org.yu E-mail: kkenpi@yunord.net

Klub je osnovan 01.08.1999. godine. Osnivač i glavni trener u klubu je Milorad Ćopić, majstor karatea - crni pojas 4. dan. Klub je član Karate federacije Srbije, najjače i najorganizovanije karate federacije u našoj zemlji.

KK ENPI je organizator jednog od najvećih turnira u Evropi – pod nazivom Super Enpi EUROPA karate kup. Na turniru 2007 godine nastupilo je oko 1800 takmičara iz 100 klubova iz 10 država.

Rezultati kluba u periodu 01.09.1999. – 22.12.2005.

KK Enpi je nastupio na 428 takmičenju i osvojio 3671 medalja.

Period	Zlato	Srebro	Bronza	Ukupno
01.09.1999. – 31.12.1999.	46	44	36	126
01.01.2000. – 31.12.2000.	97	87	72	256
01.01.2001. – 31.12.2001.	203	133	104	440
01.01.2002. – 31.12.2002.	260	181	134	575
01.01.2003. – 31.12.2003.	237	188	124	549
01.01.2004. – 23.12.2004.	248	160	146	554
01.01.2005. – 22.12.2005	287	161	162	610
01.01.2006. – 31.12.2006	**274**	**139**	**148**	**561**
Ukupno osvojeno	**1652**	**1093**	**926**	**3671**



Slika 95: Šampionski kata tim