www.elitesecurity.org

FORUM: VB & ASP

# Korak po korak: VB 6, Access, ADO i SQL.

## *Sadržaj*



Pre nego se upustimo u pojašnjenje teme da navedem koji sistem ja imam kako bi se izbegli bar neki problemi:

- Visual Basic 6 + Service Pack 5 (SP5 potražite na www.microsoft.com/vstudio)
- Access 2000
- Windows 2000 + Service Pack 4
- ADO 2.5, 2.5 Service Pack 2, 2.6, 2.7 (odgovarajući ADO/MDAC skinite sa www.microsoft.com/data)

Sve u svemu, instalirajte Visual Basic 6 i stavite Service Pack 5, instalirajte Access 2000 (ili noviji) kao i ADO 2.5 Service Pack 2 i ne bi trebalo da imate problema sa praćenjem teksta. Savetujem i da pređete na Windows 2000 (Professional ili Server) ili XP Professional.

# 1. Pravljenje proste baze podataka u Access-u 2000

Ideja je sledeća: treba da napravimo bazu podataka u kojoj ćemo držati podatke o korisnicima (username, password, ime, prezime, Datum rođenja, telefon). Dalje, recimo da se radi o nekom sajtu gde korisnici mogu da postavljaju razne tekstove. Dakle, pored podataka o korisnicima moramo da u bazi držimo podatke o tekstovima koje korisnici pišu (sam tekst, korisnik koji je napisao tekst i još i vreme kada je tekst postavljen na sajt, tj. u bazu). Slično kao što EliteSecurity.org ima podatke o korisnicima i porukama. Ništa komplikovano.

Kako vidimo, imamo dve celine: korisnici (i podaci o njima) i tekstovi i dodatni podaci o tekstu (koji korisnik je pisao tekst, vreme kada je tekst postavljen u bazu). Prema ovome, dolazimo do zaključka da ćemo u bazi podataka da imamo dve tabele: jedna je tabela koja sadrži podatke o korisnicima, a druga o tekstovima.

Prva tabela sa podacima o korisnicima imaće minimalno ova polja (a kasnije ćemo dodati još jedno polje):

```
Username    Password    Ime     Prezime     DatumRodjenja   Telefon
-------------------------------------------------------------------------------
degojs      abc123      Dejan   Gojsević    27. mart 1974.  (519) 880-1234
jc denton   jcjc        Neša    Denton      08. mart 1954.  (041) 111-1234
```

Druga tabela sa podacima o tekstovima imaće minimalno ova polja (i ovde ćemo dodati još jedno polje posle):

```
UserID      Tekst                                   Datum
-------------------------------------------------------------------------------
1           Prvi tekst korisnika sa UserID = 1      21-11-2003 14:14:00
```

Polje Datum u ovoj tabeli jeste datum i vreme kada je tekst postavljen u bazu podataka.

Dakle, prvi korak bi bio pravljenje potrebne baze podataka..

## *1.1. Pravljenje tabele tbl_Korisnici*

Pokrenite Access i odaberite opciju da kreirate novu praznu bazu podataka (Blank database). Fajl u kom će biti smešteni podaci dajte ime "db1.mdb".
Podaci u bazama podataka poput Access-a drže se u tabelama. Tabele sadrže kolone (ili polja) koje su nekako međusobno povezane logički. Da vidimo prvu tabelu koju ćemo kreirati:

- tabela "tbl_Korisnici" sa unetim podacima o nekoliko korisnika bi izgledala ovako:

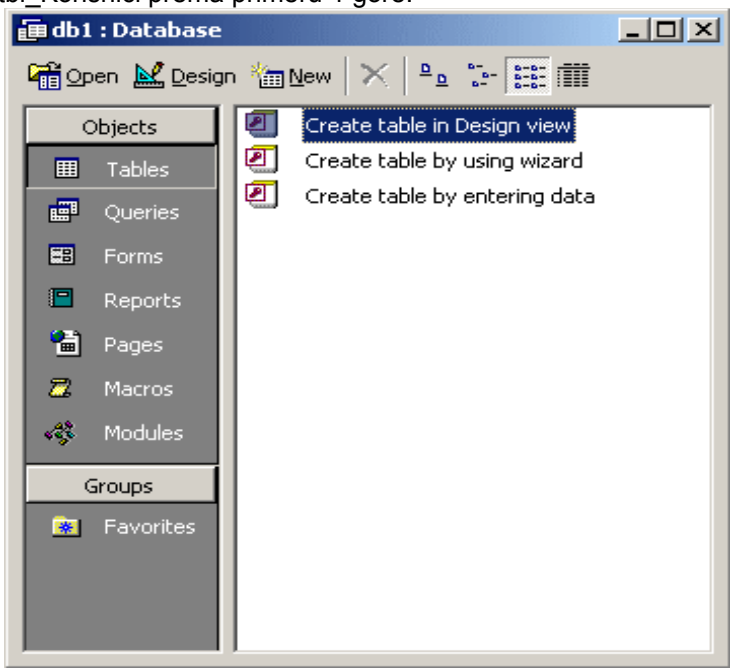
```
UserID  Username    Password    Ime      Prezime      DatumRodjenja   Telefon
-------------------------------------------------------------------------------
1       degojs      abc123      Dejan    Gojsević     27. mart 1974.  (519) 880-1234
2       jc denton   jcjc        Neša     Denton       08. mart 1954.  (041) 111-1234
3       žile        žile1       Žika     Slika        11. jun  1991.  (041) 112-1234
```

**primer 1 – primer tabele sa unetim podacima**

Kao što vidimo, u našoj tabeli "tbl_Korisnici" imamo sledeće kolone (polja):
- UserID
- Username
- Password
- Ime
- Prezime
- DatumRodjenja
- Telefon

Da odmah razjasnimo i još neke pojmove:
- TABLE – je tabela, skup svih zapisa, kao što vidite gore na primeru 1. U bazi možemo imati mnogo tabela.
- RECORD – je jedan zapis (tj. jedan red), u tabeli, npr.:
  `1  degojs  abc123  Dejan  Gojsević  27. mart 1974.  (519) 880-1234`
- RECORDSET – je nekoliko redova u tabeli, tj. nekoliko zapisa. Npr.:
  `2  jc denton  jcjc  Neša  Denton  08. mart 1954.  (041) 111-1234`
  `3  žile  žile1  Žika  Slika  11. jun 1991.  (041) 112-1234`
- FIELD – je polje ili kolona u zapisu ili tabeli. Npr. Username je polje. Dakle jedan zapis ili tabela ima više polja (npr. UserID, Username, Password.. su polja)

Vreme je da kreiramo tabelu tbl_Korisnici prema primeru 1 gore:




**slika 1: "početni prozor" u Access-u**

1. Kliknite na "Tables" stavku levo u početnom prozoru (vidi sliku 1)
2. Kliknite 2 puta na "Create table in design view"
3. Otvorio se Design prozor gde unosimo podatke o kolonama (Fields) buduće tabele

4. U kolonu Field Name unesite UserID
5. U kolonu Data Type unesite Autonumber
6. To je to. Kreirali smo kolonu UserID.
7. Prelazimo u sledeći red, u kolonu Field Name unesite Username
8. U kolonu DataType stavite Text, a u donjem delu za Field Size stavite recimo 20. Ovim smo postavili ograničenje od 20 znakova za Username.
9. Takođe, promenite Required u YES što bi značilo da je polje obavezno - svaki zapis mora imati uneto Username.

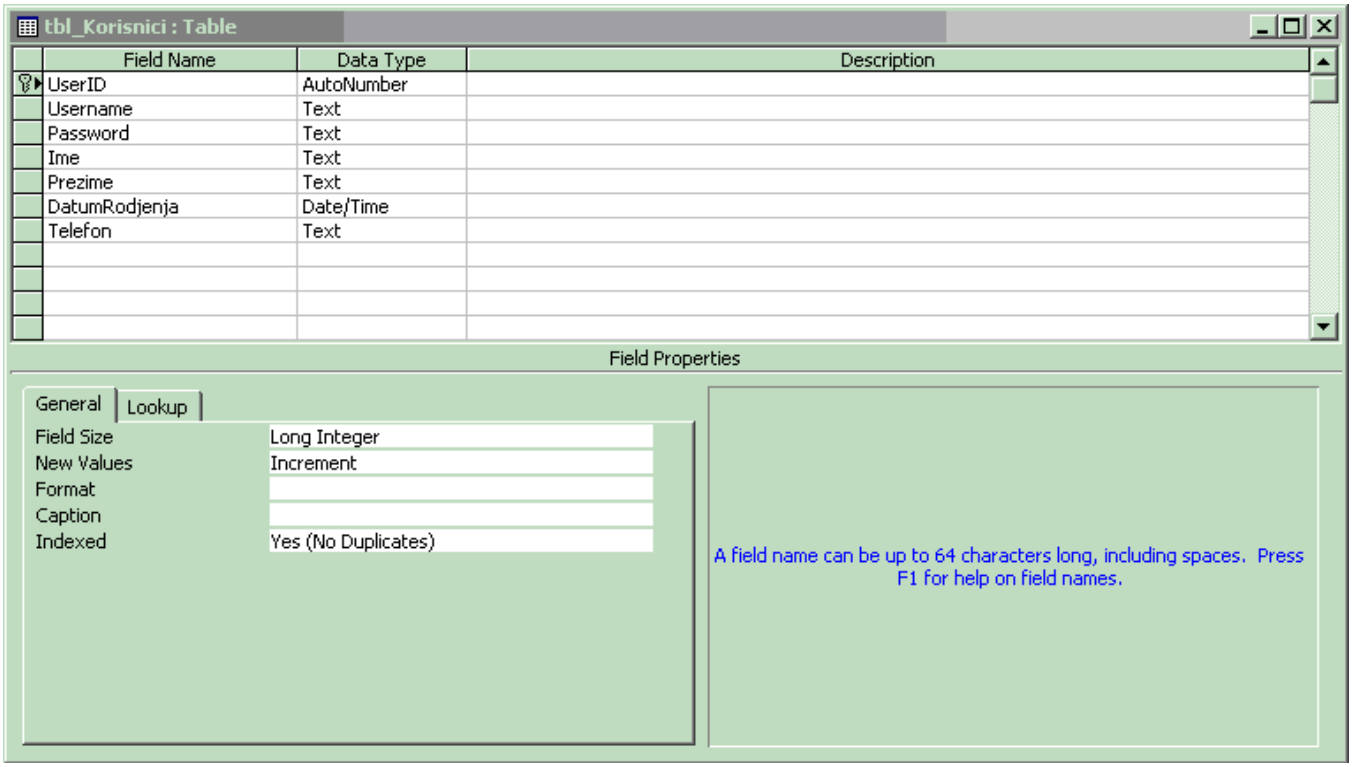
Nastavite dalje dok ne unesete podatke za sva polja koja ćemo imati u tabeli:

```
Field Name      Data Type       Field Size    Required
-------------------------------------------------------
UserID          AutoNumber      Long Integer
Username        Text            20            YES
Password        Text            20            YES
Ime             Text            15
Prezime         Text            15
DatumRodjenja   DateTime
Telefon         Text            10
```

Kao što vidite, Username i Password imaju Required atribut postavljen na YES što bi značilo da to polje prilikom unosa podataka mora biti popunjeno, tj. ne može biti prazno. Polje UserID smo postavili na AutoNumber što znači da će baza podataka sama upisivati podatke u ovo polje kako mi budemo dodavali nove korisnike u bazu. Dakle, ovo polje ne popunjavamo mi, nego sama baza. Videćemo malo kasnije to na delu.

Sad lepo kliknite desnim dugmetom na red UserID i odaberite opciju PRIMARY KEY.
Nakon ovog, polje (kolona) UserID je označeno kao PRIMARY KEY polje u ovoj tabeli. Za sada ostavite kako jeste, a kasnije ćemo objasniti šta je to Primary Key.

Evo i slika ovde da vidimo otprilike kako bi to izgledalo:



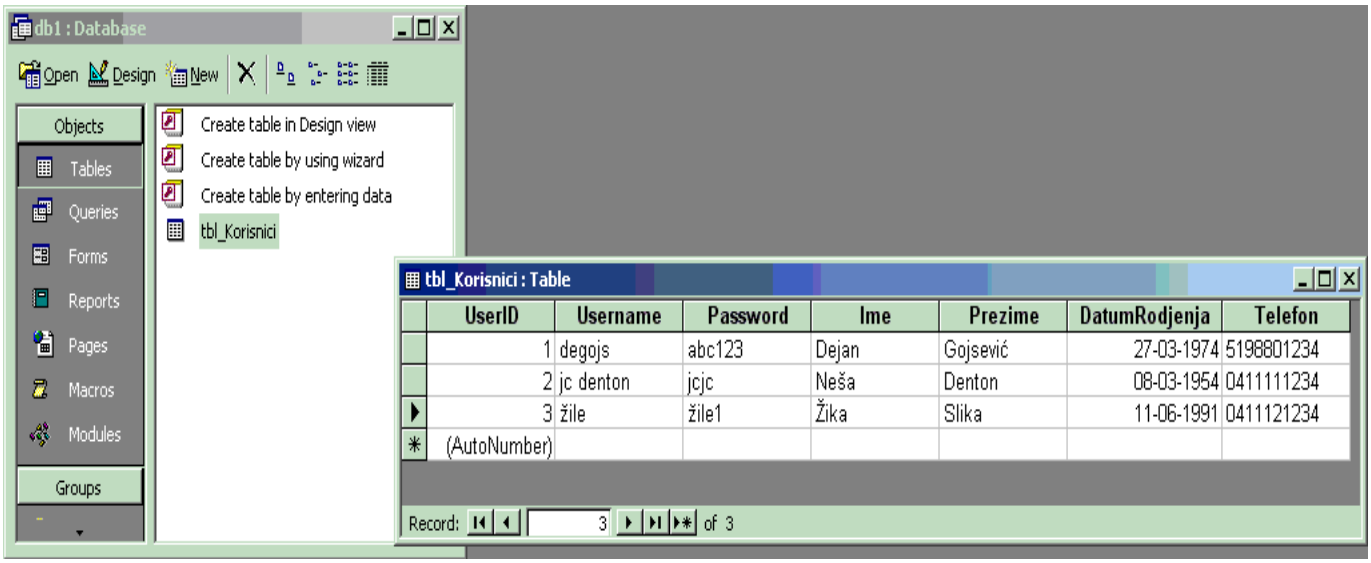

Nakon što smo kreirali bazu hajde da unesemo neke podatke u nju. Pošto ima još posla oko organizovanja baze podataka, podatke ćemo unositi koristeći sam Access, a kasnije, u trećem delu, ćemo da vidimo kako da to radimo iz našeg VB programa (ili ASP skripte na sajtu). Za sada, u tabelu koju smo upravo napravili unećemo sledeće podatke:

| UserID | Username | Password | Ime | Prezime | DatumRodjenja | Telefon |
|---|---|---|---|---|---|---|
| 1 | degojs | abc123 | Dejan | Gojsević | 27. mart 1974. | (519) 880-1234 |
| 2 | jc denton | jcjc | Neša | Denton | 08. mart 1954. | (041) 111-1234 |
| 3 | žile | žile1 | Žika | Slika | 11. jun 1991. | (041) 112-1234 |

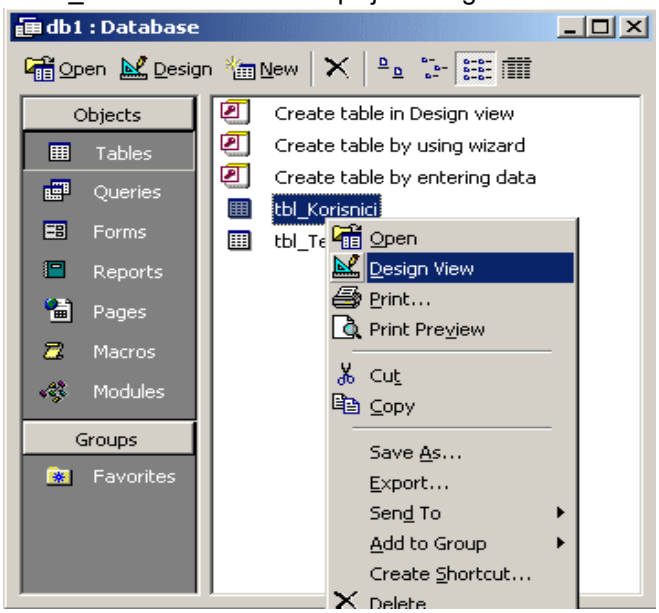
**primer 2: podaci u tabeli tbl_Korisnici**

Zatvorite Design prozor i dobićete pitanje da li želite da sačuvate tabelu i kako da je nazovete. Unesite tbl_Korisnici i to je to, tabela je sačuvana u fajlu db1.mdb.
Sad treba da unesemo podatke sa primera 2 u bazu. Jednostavno, u onom početnom prozoru (slika 1, gore) kliknite levo na Tables, a zatim dvoklik desno na tbl_Korisnici. Otvoriće se tabela koja je prazna, a vi lepo krenite da unosite podatke u nju. NAPOMENA: podatke u koloni UserID ne unosite – tu će sama baza da ubaci odgovarajuće brojeve – pristetite se da je tip ovog polja AutoNumber što znači da će baza sama da ubaci potrebne brojeve.

Nakon što završite sa unosom podataka trebalo bi da imate ovakvu sliku na ekranu:




Odlično, završili smo sa kreiranjem tabele tbl_Korisnici i uneli smo nekoliko zapisa u istu. Ako kojim slučajem poželite da dodate neko polje u tabelu ili izvršite neke promene – nema problema: u početnom prozoru levo kliknite na “Tables”, a zatim desnim dugmetom kliknite na tbl_Korisnici i odaberite opciju “Design View”.



## *1.2. Pravljenje tabele tbl_Tekstovi*

Za potrebe ovog uputstva kreiraćemo još jednu tabelu koja će biti povezana sa tabelom koju smo već napravili. Da ponovimo, ideja je sledeća: imamo npr. sajt na kojem korisnici mogu da se registruju (ove podatke držimo u tabeli tbl_Korisnici, to smo već napravili). Pored što mogu da se registruju, korisnici mogu da postavljaju tekstove – za to ćemo da kreiramo tabelu tbl_Tekstovi. Dakle, imaćemo 2 tabele: jedna drži podatke o korisnicima, a druga tekstove koje su korisnici ostavili na sajtu (slično kao što ostavljete poruke na EliteSecurity.org).

Da krenemo sa kreiranjem druge tabele, tbl_Tekstovi: u početnom prozoru levo kliknite “Tables”, a zatim 2 puta desno na “Create table in design view”. Potom kreirajte sledaća polja u tabeli:

```
Field Name      Data Type       Field Size      Required
-----------------------------------------------------------
TekstID         AutoNumber      Long Integer
UserID          Number          Long Integer    YES
Tekst           Memo                            YES
Datum           DateTime                        YES
```

Polje “TekstID” postavite da bude Primary Key. Polje “Tekst” smo stavili da je tipa Memo što u biti znači da možemo u njega da stavimo do 65536 znakova (polje tipa Text omogućava da se u njega stavi do 255 znakova što može biti malo). Polje “UserID” ćemo povezati sa tabelom tbl_Korisnici što znači da će svaki tekst MORATI da ima pridruženog korisnika iz tabele tbl_Korisnici. Slično kao što na ES poruke mogu da ostavljaju registrovani korisnici, isto tako u našu tabelu tbl_Tekstovi, tekstove mogu da postavljaju samo korisnici koji su upisani prethodno u tabelu tbl_Korisnici.
Sačuvajte (SAVE) tabelu pod imenom tbl_Tekstovi i unesite neke podatke u nju. Npr.:

```
TekstID         UserID      Tekst                                       Datum
-------------------------------------------------------------------------------
1               1           Prvi tekst korisnika sa UserID = 1     21-11-2003 14:14:00
2               1           Drugi tekst korisnika sa UserID = 1    22-11-2003 15:15:00
3               2           Prvi tekst korisnika sa UserID = 2     22-11-2003 16:16:00
```

Dakle, brojeve u polju TekstID ne unosite, njih će sama baza da ubaci pošto je to polje tipa AutoNumber.

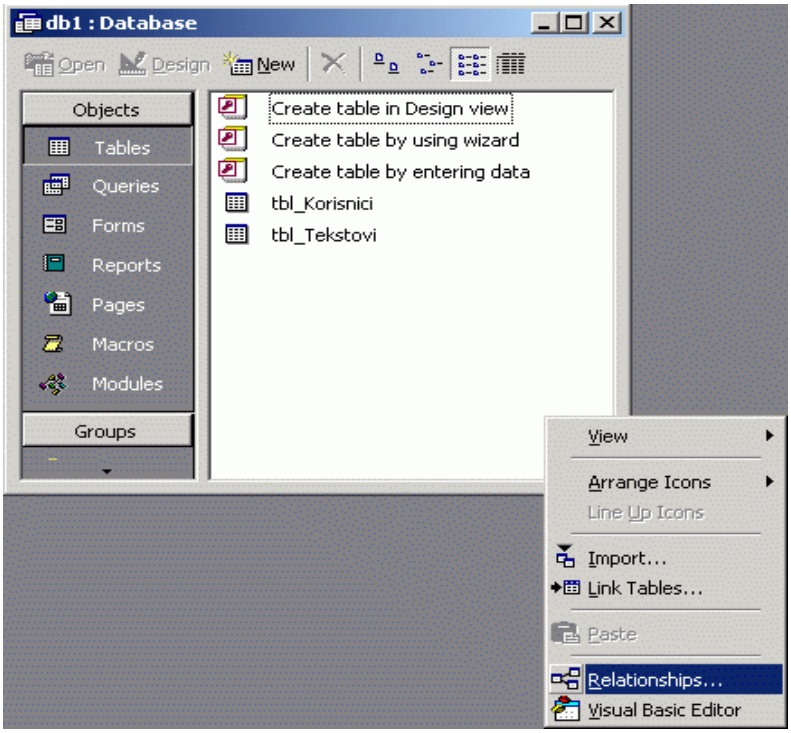
## *1.3. Povezivanje tabela - PRIMARY i FOREIGN KEY*

Primetite da prilikom unošenja podataka u tabelu tbl_Tekstovi u polje UserID možete da unesete neki broj koji NE postoji u istom tom polju u tabeli tbl_Korisnici.
Ako pokuštate da unesete recimo (NEMOJTE OVO DA UNOSITE!):

```
TekstID         UserID      Tekst                                          Datum
-------------------------------------------------------------------------------
4               999         Prvi tekst korisnika sa UserID = 999   11-11-2003 11:11:12
```

ovo SADA može da se unese, ali nije li to totalno besmisleno pošto mi u tabeli tbl_Korisnici uopšte nemamo korisnika sa UserID = 999 ? Da vidimo kako to da sprečimo. Dakle, mi želimo da tekstove u tabelu tbl_Tekstovi mogu da postavljaju samo korisnici koji postoje u tabeli tbl_Korisnici. Logično, zar ne?

Potrebno je da povežemo ove dve tabele tako da UserID u tbl_Tekstovi može da bude samo neki broj koji postoji u polju UserID u tbl_Korisnici. Prvo kliknite u prazan prostor ispod one dve tabele i odaberite opciju “Relationships”, kao na slici:




Access će potom da vas upita koje tabele želite da dodate na dijagram i vi dodajte tbl_Korisnici i tbl_Tekstovi.

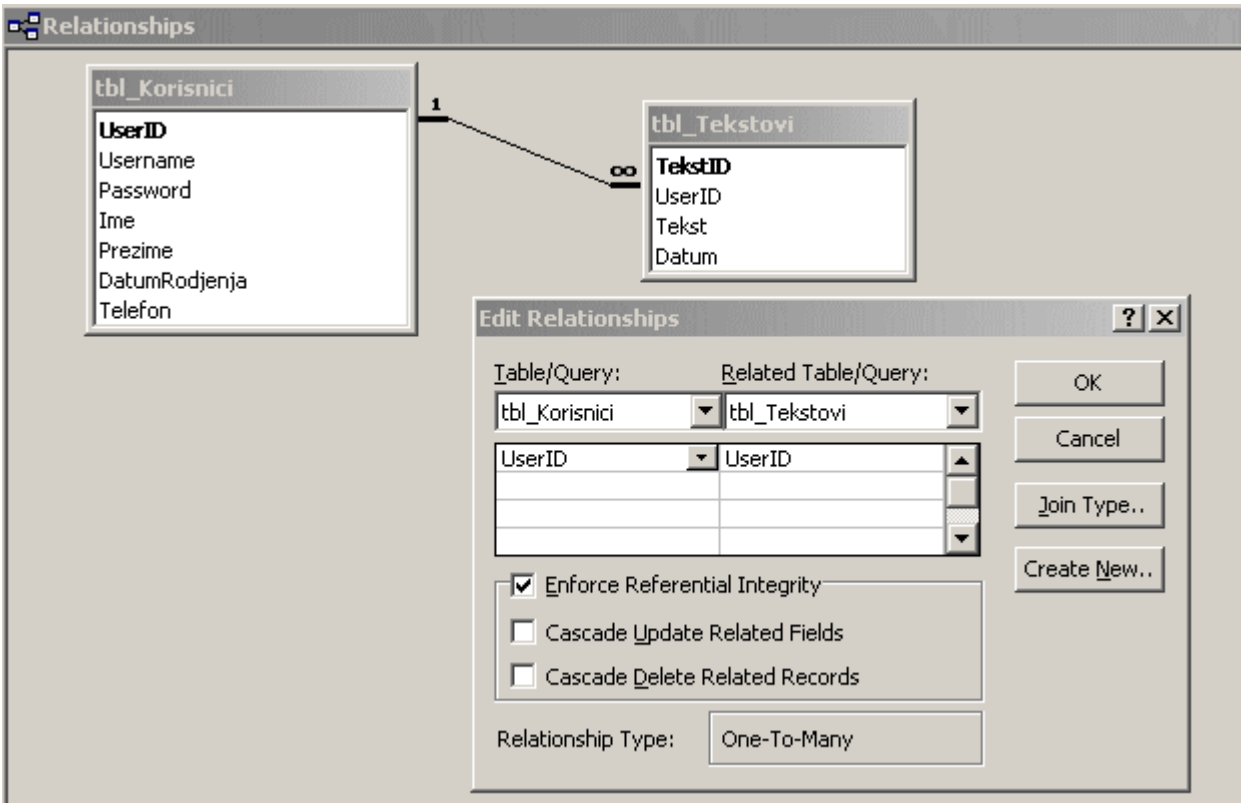
Kliknite levim dugmetom (i dalje držite pritisnuto) na polje UserID u tbl_Tekstovi prozorčiću i “prevucite” ga do polja UserID u tbl_Korisnici. Otpustite levo dugme miša iznad ovog polja i pojaviće se dijalog boks kao na slici. Postavite sve kao na slici i kliknite na OK. Ovim ste postavili vezu između polja UserID u ove dve tabele.

Primetite da će se na dijagramu pojaviti veza JEDAN prema MNOGO između ovih polja. To znači da jedan UserID (korisnik) u tabeli tbl_Korisnici može da se pojavi mnogo puta u tabeli tbl_Tekstovi. Drugim rečima, jedan korisnik može imati mnogo tekstova.Takođe, u tabeli tbl_Tekstovi, u polju UserID može se pojaviti samo neki broj koji postoji u polju UserID u tabeli tbl_Korisnici.

Ako bi sada otvorili tbl_Tekstovi za unos i pokušali da unesete:

```
TekstID         UserID      Tekst                                              Datum
-------------------------------------------------------------------------------
4               999         Prvi tekst korisnika sa UserID = 999
```

ovaj put to neće moći jer baza “vidi” da ne postoji UserID 999 u tbl_Korisnici i sprečava vas da unesete nepostojeći UserID u tabelu tbl_Tekstovi. Lepo od baze, zar ne?




Polje UserID u tbl_Korisnici je PRIMARY KEY.
Polje UserID u tbl_Tekstovi je FOREIGN KEY.

*Ako pogledate jednu i drugu tabelu u Design view-u primetite da su polja UserID u obe tabele istog tipa (Data type = AutoNumber u prvoj, tj. Number u drugoj tabeli, a u obe tabele **Field size = Long Integer**) !!!*
*AutoNumber u prvoj tabeli znači da je polje tipa Number ali će baza sama da ga upisuje. U drugoj tabeli polje UserID neće upisivati sama baza već mi (zavisno od toga koji korisnik postavi tekst) pa nije Auto već samo Number.*

PRIMARY KEY polje ne može imati duplikate. Tj. u tabeli Korisnici ne mogu dva korisnika imati isti UserID, što je i logično, a zato smo ga i postavili da bude AutoNumber – ovako sama baza dodeljuje vrednosti tom polju, umesto da mi to radimo – sprečava se mogućnost da unesemo istu vrednost 2 ili više puta. Pored toga, polje koje označimo kao PRIMARY KEY se automatski indeksira, pa je pretraživanje po tom polju brzo.

Dakle, polje UserID u tbl_Korisnici smo prilikom dizajna tabele označili kao PRIMARY KEY kako bi onemogućili da dva korisnika imaju isti UserID i kako bi polje bilo indeksirano. Pošto je nemoguće da imamo duple vrednosti u ovom polju, onda možemo da ga lepo povežemo sa UserID u tabeli Tekstovi.

Primetite da polje UserID u tabeli Tekstovi NIJE primary key već FOREIGN key - logično, jer jedan isti korisnik može imati više tekstova. Polje koje je foreign key se povezuje sa poljem koje je primary key.
Da se prisetimo šta smo uneli u tabelu Tekstovi:

```
TekstID         UserID      Tekst                                          Datum
-------------------------------------------------------------------------------------
1               1           Prvi tekst korisnika sa UserID = 1     21-11-2003 14:14:00
2               1           Drugi tekst korisnika sa UserID = 1    22-11-2003 15:15:00
3               2           Prvi tekst korisnika sa UserID = 2     22-11-2003 16:16:00
```

Dakle, UserID u tbl_Tekstovi može imati duple vrednosti (imamo dva puta UserID = 1) što je OK. Ovaj korisnik ima dva teksta sačuvana u ovoj tabeli. Isto kao što i vi možete da postujete mnogo poruka na ES.

## *1.4. Pravljenje query-ja qry_Korisnici*

Query (u nekim drugim bazama Query se naziva *View*) služi za "pregledanje" tj. odabir podataka iz jedne ili više tabela. Recimo želimo da dobijemo listu svih korisnika, ili listu svih korisnika koji su rođeni pre 1980, ili želimo da nađemo sve tekstove koje je napisao određeni korisnik.
Da bi se dobili ti podaci mogu da se koriste dve stvari:

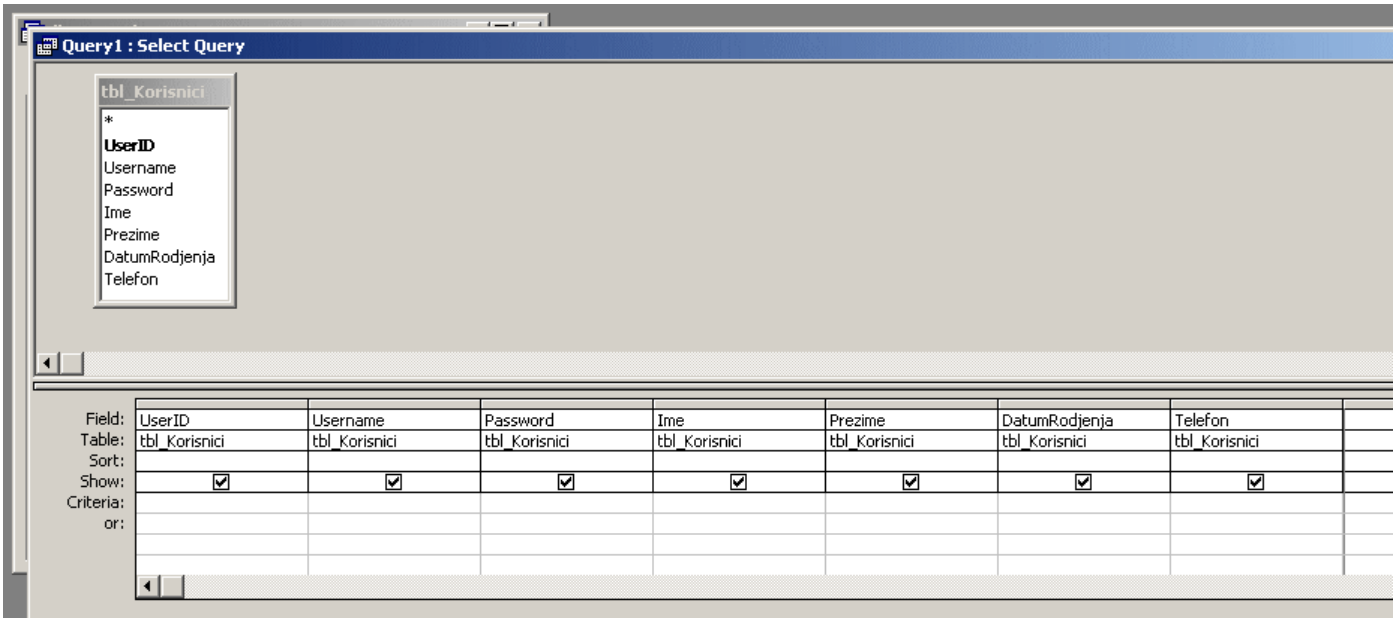
- Podaci se dovlače direktno iz tabela
- Podaci se dovlače iz Query-ja koji je unapred definisan i koji dovlači podatke iz tabela.

Postoji mnogo razloga zašto je bolje koristiti Query-je nego tabele za ovakve potrebe (npr. sigurnost – korisniku dajemo mogućnost da čita Query ali ne i tabelu, a Query je unapred definisan – neka polja iz tabele ne moraju ni da se pojave u Queryiju kao npr. polje Password, itd) tako da ćemo i mi ići tim proverenim putem. U drugom delu koji se odnosi na osnove SQL-a, napomenućemo kako se prave View-i u bazama podataka poput Oracle i SQL Server.

Elem, idemo da napravimo qry_Korisnici, a potom qty_Tekstovi.

U "početnom prozoru" Access-a, kliknite levo na Queries i potom desno 2x kliknite na Create Query in Design View. Access će da vas pita koje tabele želite da koristite u Queryiju i vi dodajte SAMO tabelu tbl_Korisnici a potom kliknite na Close.

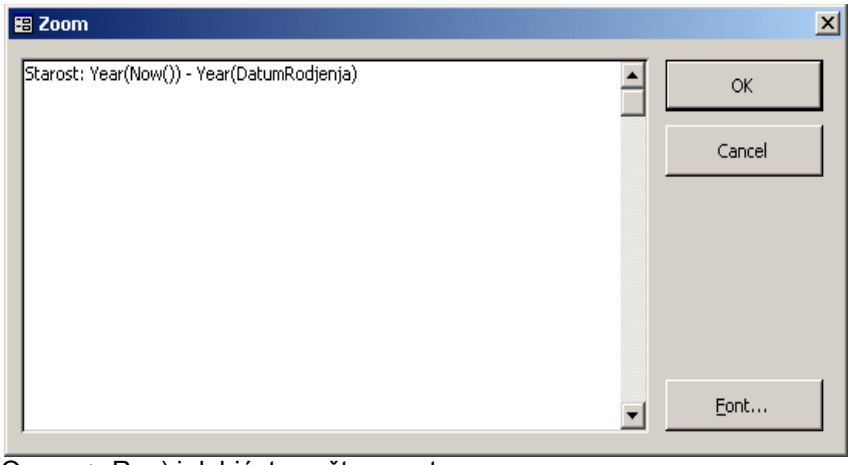
Sada je potrebno da odaberete koja polja će da se pojave u vašem kveriju. Pa, recimo da želite da imate sva polja iz tabele. Jednostavno u gornjem delu kliknite 2x na polje UserID, zatim 2x na polje Username i tako redom na sva polja. Dobićete nešto kao na ovoj slici dole:




Sada treba da vidimo koje podatke ovaj kveri daje kada se pokrene. Idite na meni QUERY i odaberite opciju RUN. Dobićete praktično iste podatke kao kada otvorite tabelu. Dobro, 'ajde da se vratimo nazad u Query Design i da napravimo neke izmene: kliknite na VIEW i odaberite opciju DESIGN VIEW (usput, možete da odaberete i View->SQL View pa ćete da vidite SQL izraz koji je u stvari Access kreirao za ovaj Query).
U stvari, kad razmislim malo, šta će vam godina rođenja? Zar ne bi bilo bolje da umesto godine rođenja imate podatak koliko godina ima korisnik? Starost ćemo izračunati jednostavno prema formuli:
Starost = sadašnja godina – GodinaRodjenja

Hajde da vidimo kako da promenimo kveri pa da dobivamo starost korisnika umesto godine rođenja:

- U donjem delu Design prozora, tamo gde imate navedena polja, kliknite jednom u ćeliju gde piše "DatumRodjenja" (levo od ćelije "Prezime", desno od "Telefon" i iznad "tbl_Korisnici") i pritisnite SHIFT+F2 i otvoriće se jedan prozor sa imenom "Zoom" u koji ukucajte:
  Starost: Year(Now()) - Year(DatumRodjenja)




E sada pokrenite kveri (Query -> Run) i dobićete nešto poput:

Query1 : Select Query

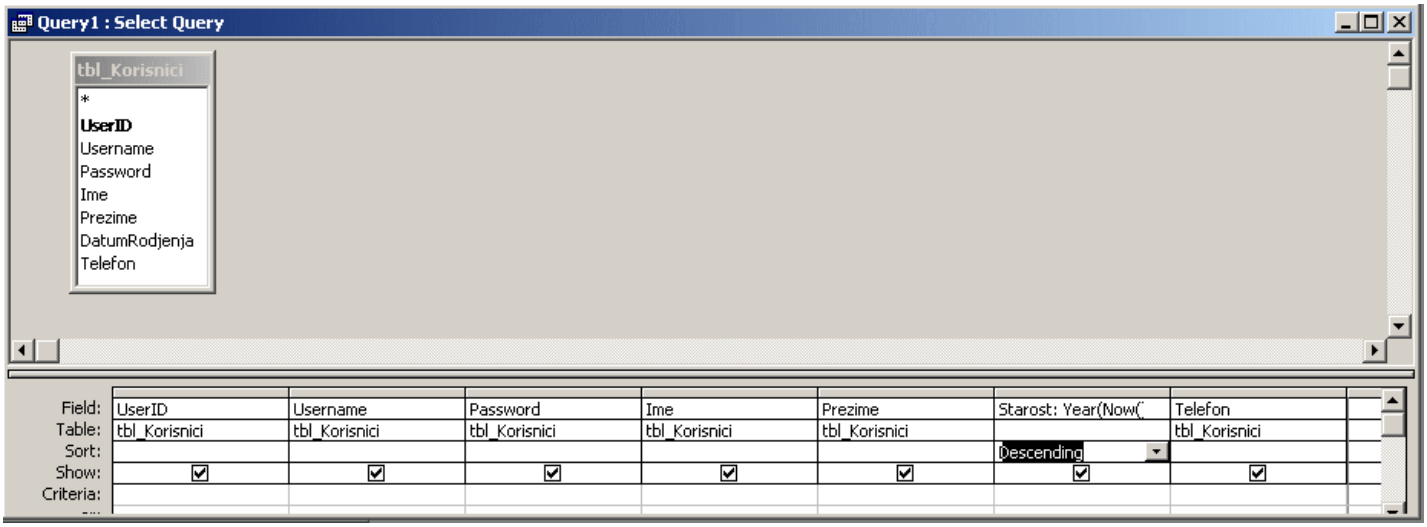
| UserID | Username | Password | Ime | Prezime | Starost | Telefon |
|---|---|---|---|---|---|---|
| 1 | degojs | abc123 | Dejan | Gojsević | 29 | 5198801234 |
| 2 | jc denton | jcjc | Neša | Denton | 49 | 0411111234 |
| 3 | žile | žile1 | Žika | Slika | 12 | 0411121234 |
| (AutoNumber) | | | | | | |

Record: 1 of 3

Primetite da više nemate kolonu DatumRodjenja već kolonu Starost u kojoj se lepo izračunate godine za svakog korisnika. Da malo pojasnimo kako radi ona formula:
Starost: Year( Now() ) - Year ( DatumRodjenja )

Dakle “Starost” je naziv “nove” kolone. Dalje, trenutnu godinu dobivamo pomoću Year( Now() ).
Now() vraća današnji datum uključujući i dan i mesec, ali pošto nas zanima samo godina onda imamo Year( Now() ).
Pošto je sada godina 2003, ova formula vraća 2003.
Dalje, imamo Year( DatumRodjenja ). DatumRodjenja je podatak koji smo uneli za svakog korisnika i za konkretno korisnika degojs uneli smo 27-03-1974 što je moj datum rođenja. Pošto nam je potrebna samo godina (1974) onda koristimo opet ono Year( DatumRodjenja ) što će za slučaj korisnika degojs da vrati 1974.
Na kraju oduzmemo jedno od drugog i u mom slučaju imamo:
Starost: 2003 – 1974
što ispravno daje 29. I tako dalje za svakog korisnika. Lepo, zar ne..

E sad, bilo bi lepo kada bi ovi korisnici bili sortirani po starosti: od najstarijeg do najmlađeg. Evo kako to da uradite:
vratite se u Design pogled (View -> Design view) i u koloni Starost postavite polje SORT na DESCENDING kao na slici:




Pokrenite sada kveri (Run -> Query) i dobićete nešto slično kao pre ALI videćete da su zapisi sortirani prema polju "Starost" i to od najstarijeg prema najmlađem. Lepo, zar ne? :-)

Query1 : Select Query

| UserID | Username | Password | Ime | Prezime | Starost | Telefon |
|---|---|---|---|---|---|---|
| 2 | jc denton | jcjc | Neša | Denton | 49 | 0411111234 |
| 1 | degojs | abc123 | Dejan | Gojsević | 29 | 5198801234 |
| 3 | žile | žile1 | Žika | Slika | 12 | 0411121234 |
| (AutoNumber) | | | | | | |

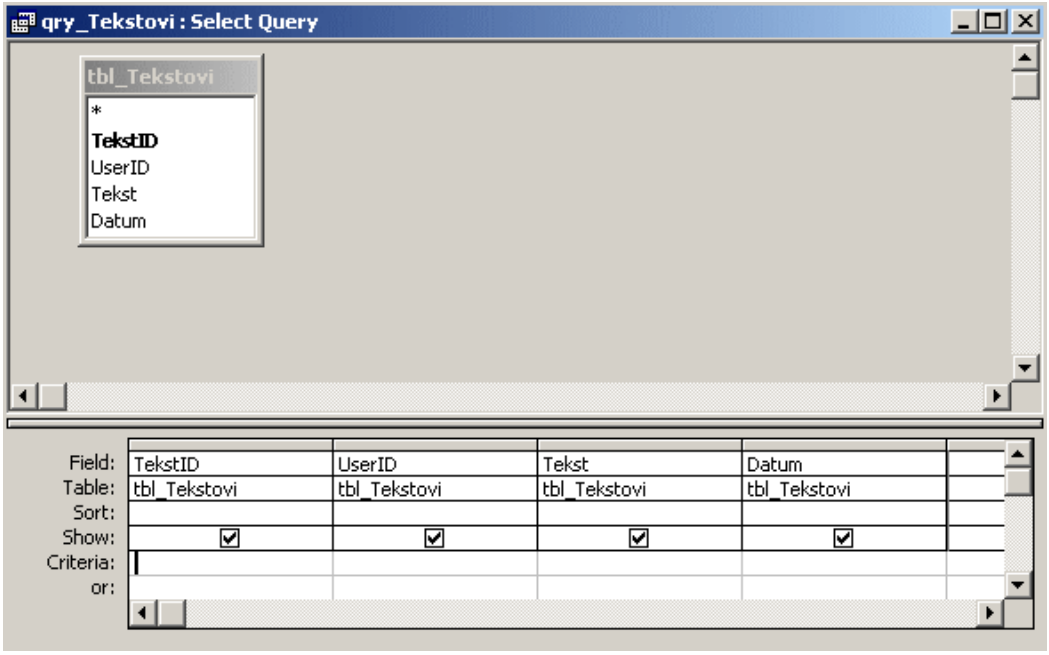
Record: 1 of 3

Naravno, možete malo da se igrate i sortirate rezultat po raznim poljima, ili po više njih. Možda nije loše otići i na View -> SQL View i videti kako Access menja SQL izraz.

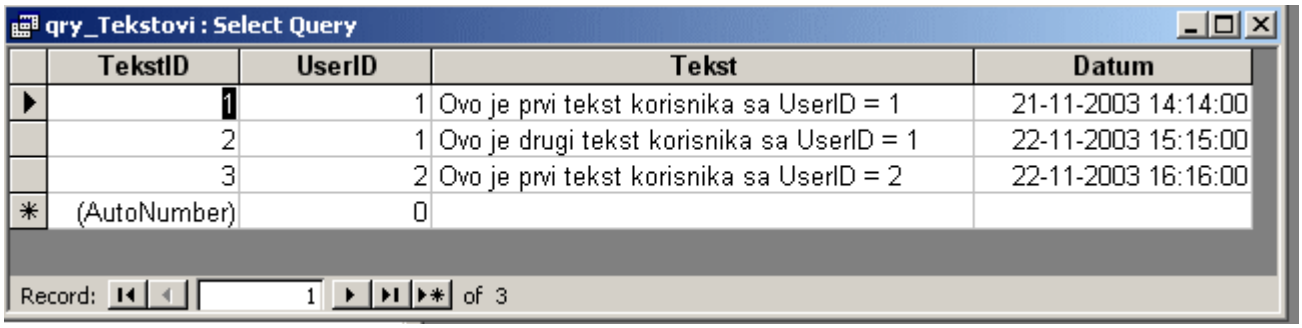
Dooooo-bro, hajde da sada napravimo i onaj drugi kveri – qry_Tekstovi.

### *1.4. Pravljenje query-ja qry_Tekstovi*

Ovaj kveri ćemo koristiti da dobijemo sve tekstove koji postoje u tabeli tbl_Tekstovi. Jednostavno, u "početnom prozoru" kliknite levo na Queries, a onda desno kliknite 2x na "Create Query in Design View". Dodajte tabelu tbl_Tekstovi i zatim dodajte sva polja iz tabele u query, kao na slici:

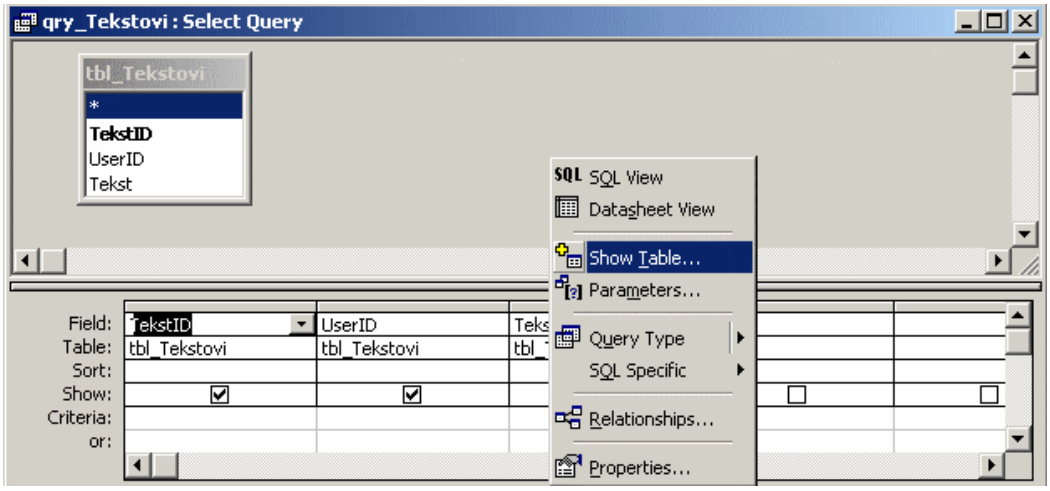



Kada pokrenete ovaj Query (Query -> Run) dobićete ovakav rezultat:

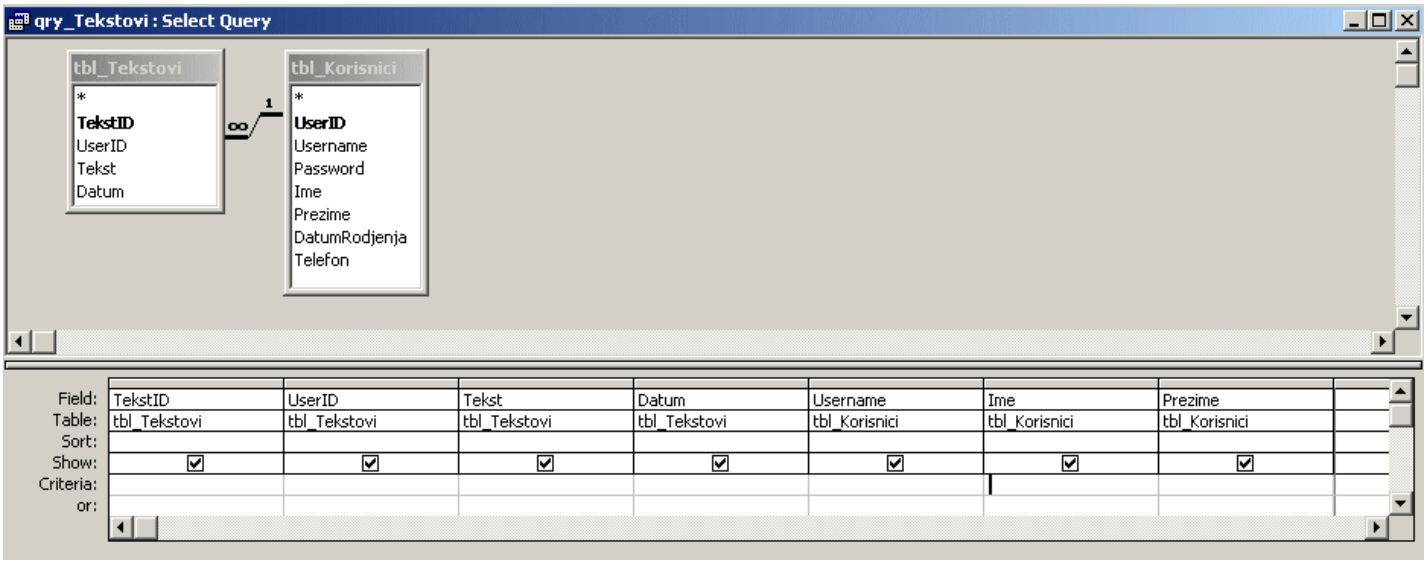



Lepo.. pa i nije baš :-) Umesto što imamo UserID polje i u njemu brojeve 1, 1 i 2, zar ne bi bilo lepše da dobijemo korisničko ime korisnika koji je pisao tekst? Dakle pored UserID da imamo recimo Username polje i da tu lepo piše "degojs" i "jc denton". Ili možda čak polja "Ime" i "Prezime"? Da vidimo kako ćemo to da uradimo.

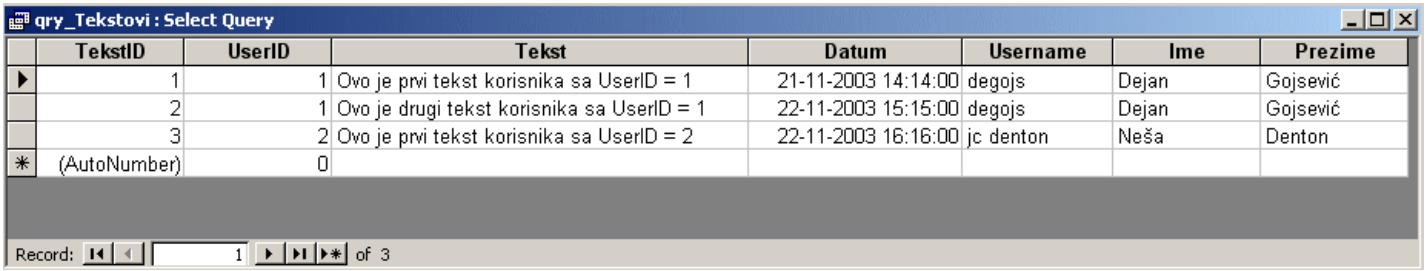
- Vratite se u Design View (View -> Design View)
- Kliknite desnim dugmetom u prazan prostor gore pored tabele tbl_Tekstovi i odaberite Show Table opciju:




- Dodajte tabelu tbl_Korisnici i zatvorite taj prozorčić. Sada bi trebalo da imate ispred sebe ovakvu situaciju:
- E sad lepo kliknite 2x na sledeća polja u tabeli tbl_Korisnici: Username, Ime, Prezime i imaćete ovakvu situaciju na ekranu (odmah vidimo i vezu između polja UserID u jednoj i drugoj tabeli):




Sada pokrenite kveri ( Query -> Run ) i dobićete ono što smo obećali, sve tekstove, ali ovaj put pored UserID polja, imate i polja Username, Ime, Prezime. Lepo, zar ne? :-)




Zatvorite Design view (sačuvajte kveri pod imenom qry_Tekstovi) i mi smo završili sa prvim delom ovog uputstva.

Da se podsetimo šta smo naučili:

- Kreiranje tabele
- Unošenje podataka u tabelu pomoću Access-a
- Povezivanje dve tabele (Primary key – Foreign Key veza JEDAN-PREMA-MNOGO). Ovakav odnos dve tabele još se naziva i PARENT-CHILD odnos.
- Kreiranje kverija pomoću jedne i više tabela.

Naravno, sad malo možete da sami “čačkate” i probavate: dodajte još neka polja, kreirajte još poneki kveri (query = eng. upit, jer mi pitamo bazu da nam da neke podatke :-), sortirajte po raznim poljima, itd, itd.

Bilo bi dobro da ovo štivo savladate dobro pre nego što krenemo da objašnjavamo drugi deo – SQL.

Uz ovaj tekst imate i Access 2000 bazu “db1.mdb” pa razgledajte, probavajte i – pitajte!

Pozdrav,
Dejan “degojs” Gojsević

www.elitesecurity.org

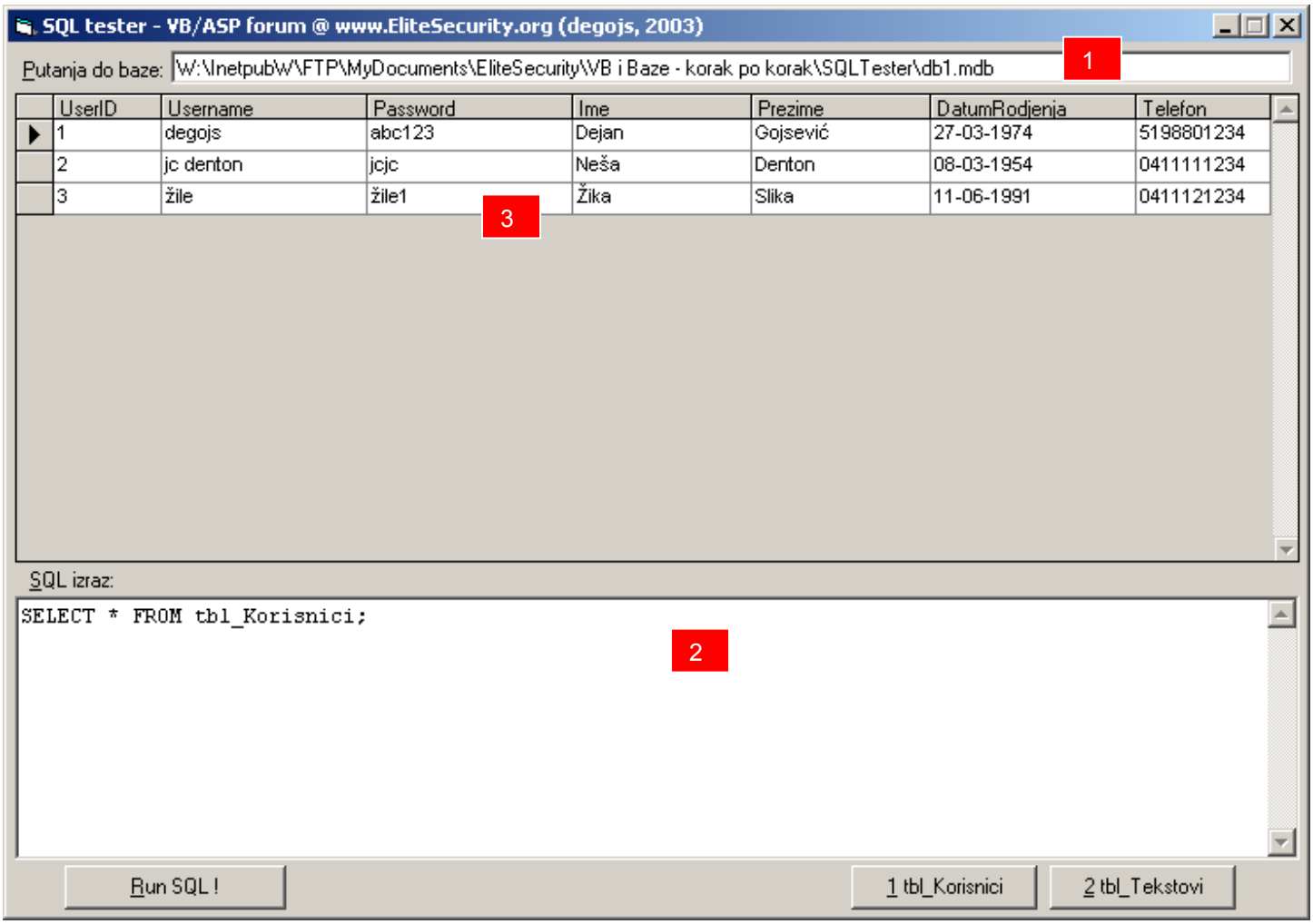
VB & ASP FORUM

# Korak po korak: VB, Access, SQL i ADO

## 2. Osnove SQL-a

Drugi deo uputstva objasniće sitaksu SQL jezika koji se koristi pri radu sa velikim brojem baza podataka raznih proizvođača. Da bi uspešno pratili ovo uputstvo, potrebno je da preuzmete program "SQL Tester" koji smo napisali očas posla za potrebe ovog teksta. Programčić je napisan u VB 6 i pored izvršne (exe) verzije, imate na raspolaganju i čitav VB projekt pa tako možete da vidite i izvorni kod ako ste sumnjičavi :-) ili ako želite da malo menjate način na koji program radi. Program je takav kakav je, ništa naročito, napisan "na brzaka" – ali ipak je najvažnije da radi posao lepo i sasvim je odgovarajući za ono čemu je namenjen. Svakako, kad proučite ovo uputstvo, možete da ga i poboljšate (u tom slučaju vas molimo, ali ne i obavezujemo, da negde navedete da ste prvobitnu verziju preuzeli sa www.EliteSecurity.org).

Kako se radi sa programom? Mislim da i ne možete da pogrešite, ali ukratko:



U delu koji je označen sa [1] unesite putanju do vašeg mdb fajla koji smo kreirali u prvom delu.
Deo koji je označen na slici sa [2] služi za unošenje raznih SQL izraza, i, nakon što ovde unesete određeni SQL izraz, kliknite na dugme "Run SQL !" da bi u delu [3] dobili rezultat u jednoj tabeli koja služi za prikaz podataka (DataGrid kontrola).
Na primeru na slici, vidite unesen SQL izraz "SELECT * FROM tbl_Korisnici;" i nakon što smo kliknuli na "Run SQL" dugme u delu [3] vidimo da su izlistani svi podaci iz tabele tbl_Korisnici. Dakle, prilično jednostavno korišćenje programčića, baš kao što i treba da bude, a opet, omogućava da lepo vežbamo SQL.

Postoje još dva dugmića na formi: "1 tbl_Korisnici" i "2 tbl_Tekstovi" i kliktanjem (odnosno pritiskom na ALT+1 ili ALT+2) na njih dobićete prikaz podataka u odgovarajućim tabelama, čisto da vas poštedim stalnog kucanja odgovarajućih SELECT izraza.

Ne zaboravite: program je testiran na sistemu na kome je instaliran Service Pack 5 za Visual Studio, kao i ADO 2.5 SP3; molim da pre prijavljivanja nekih grešaka budete sigurni da i vi imate iste stvari. Čisto da ne gubimo vreme natežući se hoće li to ili neće da proradi..

Dakle kada pokrenete program i uverite se da radi, nema šta više da se čeka – da vidimo šta i kako ..

## *2.1. SELECT (ORDER BY, WHERE, IN, TOP, DISTINCT)*

SELECT komanda se koristi za čitanje podataka iz baze. Da vidimo odmah na primeru:

```
SELECT ime FROM tbl_korisnici;
```

bi mogli da prevedemo “PROČITAJ KOLONU ime IZ tbl_Korisnici”. Dakle, dobivamo sve zapise koji se nalaze u koloni (polju) “ime” u tabeli tbl_Korisnici. Rezultat je očekivano:

```
ime
----------
Dejan
Neša
Žika
```

Naravno, moguće je da zatražimo i više polja, nije bitno kojim redom:

```
SELECT prezime, username FROM tbl_Korisnici;
```

što nam vraća:

```
prezime          username
---------------------------------
Gojsević         degojs
Denton           jc denton
Slika            žile
```

E sad, kolona “username” i nije baš naš jezik, pa da vidimo kako da to promenimo samo za potrebe prikaza:

```
SELECT prezime, username AS korisnicko_ime FROM tbl_Korisnici;
```

što nam vraća:

```
prezime          korisnicko_ime
---------------------------------
Gojsević         degojs
Denton           jc denton
Slika            žile
```

Kako vidite, ime kolone je “promenjeno” u “korisnicko_ime” (samo za ovaj put, ništa se u tabeli nije izmenilo).

Ponekad ne znamo (ili smo malo lenji da kucamo) imena svih kolona, pa možemo da koristimo:

```
SELECT * FROM tbl_Korisnici;
```

..što će da pročita sve kolone u tabeli.

E sad kad znamo da čitamo podatke iz tabele, da vidimo kako da malo uredimo te podatke. Recimo.. da ih sortiramo prema nekom polju:

```
SELECT prezime, ime, datumrodjenja FROM tbl_Korisnici ORDER BY prezime;
```

Vidimo da je rezultat sortiran prema prezimenu od A prema Z. Možemo da sortiramo i u obrnutom redosledu od Z prema A:

```
SELECT prezime, ime, datumrodjenja FROM tbl_Korisnici ORDER BY prezime DESC;
```

Sortiranje je moguće vršiti prema više polja, npr.

```
SELECT prezime, ime, datumrodjenja FROM tbl_Korisnici ORDER BY prezime, ime;
```

U ovom slučaju zapisi će prvo biti sortirani prema polju “prezime”, a zatim prema polju “ime”. U našem primeru ovo neće imati mnogo značaja ali recimo ako bi imali tabelu u kojoj bi imali ovakve podatke:

```
Ime         Prezime
----------------------------
Dejan       Gojsevic
Milan       Gojsevic
Dragan      Gojsevic
```

Izvršavanjem “SELECT prezime, ime FROM table ORDER BY prezime” dobili bi:

```
Prezime     Ime
---------------------------
Gojsevic    Dejan
Gojsevic    Milan
Gojsevic    Dragan
```

što baš i nije dobro jer imamo situaciju da je M(ilan) pre D(ragan). Ako izvršimo dodatno sortiranje po polju “Ime” dobićemo željeni rezultat:

```
Prezime     Ime
---------------------------
Gojsevic    Dejan
Gojsevic    Dragan
Gojsevic    Milan
```

Dobro, završili smo sortiranje, hajde da vidimo kako da dobijemo samo jedno određeno polje iz baze.

```
SELECT Ime, Prezime FROM tbl_Korisnici WHERE Ime = 'Dejan';
```

Dakle, potrebno je da upotrebimo ključnu reč WHERE i onda navedemo uslov koji je potrebno da bude ispunjen. Naravno možemo da koristimo više uslova sve u kombinaciji sa AND, OR, NOT. Npr.

```
SELECT UserId, Ime, Prezime FROM tbl_Korisnici WHERE Ime = 'Dejan' OR UserId = 2;
```

Kao rezultat ovog upita dobivamo 2 zapisa: jedno gde je ispunjen uslov da je polje Ime = Dejan, a drugi zapis je onaj koji je ispunio uslov da je UserId = 2.

Sad bi već mogli da smislimo i SQL izraz kojim bi proverili da li je korisnik prilikom prijavljivanja na sistem uneo ispravno korisničko ime i lozinku:

```
SELECT Ime, Username, Password FROM tbl_Korsinici WHERE Username = ‘degojs’ AND Password
= ‘abc123’;
```

Pokretanjem ovog SQL upita dobivamo natrag jedno polje i onda lepo vidimo i koje je ime na[eg korisnika pa možemo da mu ispišemo i pozdravnu poruku koristeći to ime rađe nego username.
Ako kao rezultat upita ne dobijemo jedan zapis onda znači da kombinacija korisničkog imena i lozinke nije ispravna, te lepo saopštimo korisniku da pokuša ponovo da se uloguje.

Postoji još jedna stvar: IN. Da vidimo kako se koristi:

```
SELECT UserId, Ime FROM tbl_Korisnici WHERE UserId IN (1,3);
```

Ovim upitom dobivamo zapise gde je UserId = 1 ili UserId = 3. Uštedeli smo malo pisanja, a inače, to smo mogli da dobijemo i ovako:

```
SELECT UserId, Ime FROM tbl_Korisnici WHERE UserID = 1 OR UserId = 3;
```

Dakle, ništa posebno, ali može da bude korisno kod pisanja ugnježdenih upita, što ćemo videti kasnije u ovom tekstu.

Dalje, ponekad je potrebno da pogledamo samo nekoliko prvih zapisa koje vraća upit. Ovo se radi pomoću TOP. Ako izvršimo upit:

```
SELECT Ime, Prezime FROM tbl_Korisnici;
```

Kao rezultat dobivamo sve zapise iz tabele. Recimo da nam je potrebno samo da vidimo samo prva dva zapisa .- koristimo TOP:

```
SELECT TOP 2 Ime, Prezime FROM tbl_Korisnici;
```

Primer verovatno i nije baš slikovit, ali kad naiđete na situaciju znaćete da treba da koristite TOP opciju.

Npr. treba da dobijemo ime i prezime korisnika koji je poslednji u bazi (nakon što sortiramo korisnike po abecedi).

Prvo da vidimo sve korisnike sortirane po abecedi:

```
SELECT Ime, Prezime FROM tbl_Korisnici ORDER BY Prezime, Ime;
```

Sad ćemo da okrenemo ovu listu naopačke tako da je sortiranje od Z prema A:

```
SELECT Ime, Prezime FROM tbl_Korisnici ORDER BY Prezime DESC, Ime DESC;
```

E sad, potreban nam je samo ovaj prvi zapis sa liste pa ćemo da prošitimo upit sa jednim TOP:

```
SELECT TOP 1 Ime & ' ' & Prezime FROM tbl_Korisnici ORDER BY Prezime DESC, Ime DESC;
```

I eto, dobili smo ime i prezime korisnika koji je poslednji na listi kada je ona sortirana od A prema Z, odnosno prvi je na listi kada je sortirana od Z prema A. Možemo da malo ulepšamo ovo i ujedno da vidimo kako više polja iz tabele da stavimo u jedno u upitu:

```
SELECT TOP 1 Prezime & ', ' & Ime AS FullName FROM tbl_Korisnici ORDER BY Prezime DESC,
Ime DESC;
```

Dakle, ono što je novo jeste:

```
Prezime & ', ' & Ime AS FullName
```

što bi u prevodu značilo: u rezultatu kreiraj polje FullName i u njega stavi polje Prezime pa onda jedan zarez pa razmak pa polje Ime. Eto. Sve ove upite probavajte u SQL Testeru i posmatrajte rezultate, mnogo je lakše za praćenje :-).

I za kraj, ostaje nam još DISTINCT klauza. DISTINCT se koristi za uklanjanje duplikata, prosto rečeno. Recimo da imamo neku tabelu sa podacima:

```
Ime         Prebivaliste
-------------------------
Dejan       Novi Sad
Milan       Niš
Pera        Kragujevac
Dragan      Niš
```

I recimo da mi želimo da vidimo koja su razna prebivališta koja imamo evidentirana u tabeli. Ako probamo

```
SELECT Prebivaliste FROM tabela;
```

Rezultat upita je:

```
Prebivaliste:
----------------
Novi Sad
Niš
Kragujevac
Niš
```

Vidimo, Niš je dva puta u rezultatu, što nam nikako ne treba. Dakle, imamo “duplikat” i valjalo bi ga ukloniti. E tu uskače u pomoć DISTINCT:

```
SELECT DISTINCT Prebivaliste FROM tabela;
```

će da vrati:

```
Prebivaliste:
----------------
Novi Sad
Niš
Kragujevac
```

Što je već mnogo bolje. Posle ovo možemo i da izbrojimo (zapise) i da lepo korisniku kažemo u bazi podataka imamo evidentirana 3 različita prebivališta, umesto 4 što nikako nije tačno. Tako ako imate sajt poput EliteSecurity.org i beležite u nekoj tabeli IP adrese vaših posetilaca možete onda očas posla da vidite koliko ste poseta sa jedinstvenih IP adresa imali u toku jednog dana, nedelje, meseca i slično.

## *2.2. COUNT, MAX, MIN, SUM, AVG*

Ponekad nam nije bitno koje podatke dobivamo nazad već samo koliko zapisa je vraćeno na naš upit. Generalno, za ovakve stvari se koristi COUNT. Da vidimo:

```
SELECT COUNT(*) AS Rezultat FROM tbl_Korisnici;
```

Ovim smo dobili jedno polje “Rezultat” u kom se nalazi broj koji odgovara broju polja u tabeli. Da vidimo još neki primer istog: recimo da želimo da vidimo koliko korisnika je rodjeno posle 1. februara 1972. Evo kako bi izgledao izraz koji bi vraćao zapise iz tabele:

```
SELECT * FROM tbl_Korisnici WHERE DatumRodjenja > CDATE('1-FEB-1972');
```

Vidimo da su u pitanju 2 zapisa, tj. korisnici degojs i žile. E sad, rekli smo da je nama potrebno samo da znamo koliko korisnika ima koji su rođeni posle tog datuma, pa nema potrebe da dovlačimo kompletne zapise iz tabele, već ćemo ih samo izbrojati pomoću COUNT.

```
SELECT COUNT(*) AS Rezultat FROM tbl_Korisnici WHERE DatumRodjenja > CDATE('1-FEB-1972');
```

Rezultat je “2” što znači da 2 zapisa zadovoljavaju uslov. Dakle, u tabeli imamo dva zapisa gde je DatumRodjenja posle 1. februara 1972. Eto, nije mnogo komplikovano zar ne?

Kako vidimo ovde smo koristili i jednu funkciju, CDATE(), koja služi da konvertujemo tekst (’1-FEB-1972’) u tip DateTime pošto je naše polje DatumRodjenja tog tipa. Ova funkcija je specifična za Access, a na drugim bazama se koriste slične funkcije za konverziju – potrebno je samo malo pogledati dokumentaciju.

Važna stvar kod COUNT je da se različito ponaša ako stavite COUNT(*) i COUNT(nekoPolje). U prvom slučaju COUNT će da broji i polja koja imaju upisanu vrednost NULL.

Funkcije MAX, MIN i AVG ne mogu baš da se primene na naše tabele ali objasnićemo ih ukratko – važno je da znate da postoje, lako ćete kasnije da se snađete. Recimo da imam tabelu sa poljima i podacima:

```
Ime         Plata
---------------------
Dejan       20.00
Dragan      24.00
Milan       30.00
```

Ako bi želeli da vidimo kolika je maksimalna plata nekog radnika:

```
SELECT MAX(Plata) FROM tabela;
```

Slično vredi i za MIN funkciju. Na kraju, AVG funkcija vraća prosek (AVeraGe):

```
SELECT AVG(Plata) FROM tabela;
```

## 2.3. GROUP BY, HAVING

E sad.. ovo možda i spada u napredni SQL, pa preskačem ovu celinu. Savladajte vi ostale stvari u ovom tekstu pa ćete vrlo lako da vidite kako se koriste i ove stvarčice. Hvala na razumevanju :)

## 2.4. "Spajanje" podataka iz više tabela: ugnježđeni upiti (nested queries), JOIN

Da vidimo šta su to ugnježđeni upiti.

Recimo da hoćemo da izlistamo sve tekstove koje je pisao korisnik degojs. Prvo, da vidimo koji je UserID ovog korisnika:

```
SELECT UserID FROM tbl_Korisnici WHERE Username = 'degojs';
```

E sad lepo vidimo da je UserID jednak 1. Sada sa tim podatkom idemo u ovu drugu tabelu:

```
SELECT * FROM tbl_Tekstovi WHERE UserID = 1;
```

i kao rezultat dobivamo dva zapisa (TekstID = 1, i TekstID = 2). Lepo, ali kako da mi to uradimo sve odjednom? Dakle, dođe naš korisnik na sajt i kaže daj mi listu svih tekstova koje je do sada napisao korisnik “degojs”. Evo kako:

```
SELECT * FROM tbl_Tekstovi
      WHERE UserID = ( SELECT UserId FROM tbl_Korisnici WHERE Username = 'degojs' );
```

Kako vidimo ništa komplikovano. Prvo se izvrši ovaj unutrašnji izraz i rezultat se onda koristi prilikom izvršavanja ovog “vanjskog” izraza.

Ajde da vidimo kako bi dobili sve tekstove koje je napisao degojs i plus tome još tekstove koje je napisao jc denton.

```
SELECT UserID FROM tbl_Korisnici WHERE Username = 'degojs' OR Username='jc denton';
```

Vidimo da smo dobili dva UserID-a, 1 i 2, za svaki username po jedan. Sad bi sa ovim išli u tabelu tekstova ovako:

```
SELECT * FROM tbl_Tekstovi
      WHERE UserID IN ( SELECT UserID FROM tbl_Korisnici
                              WHERE Username = 'degojs' OR Username='jc denton' );
```

Primetite da smo ovaj put koristili IN umesto = jer unutrašnji upit može da vrati više od jednog zapisa, a u našem slučaju i vraća više od jednog. Probajte da stavite = umesto IN i dobićete lepu poruku o grešci.

Da izbrojimo koliko je tekstova napisao korisnik degojs?

```
SELECT COUNT(*) FROM tbl_Tekstovi
           WHERE UserID = ( SELECT UserID FROM tbl_Korisnici
                              WHERE Username = 'degojs');
```

Eto, to bi bili ugnježđeni upiti ukratko. Da vidimo sad kako da “spojimo” podatke iz više tabela koristeći JOIN.

Recimo da pregledamo tabelu tekstova, ali hteli bi da umesto UserID odmah lepo imamo i korisničko ime autora teksta. Naime ako uradimo:

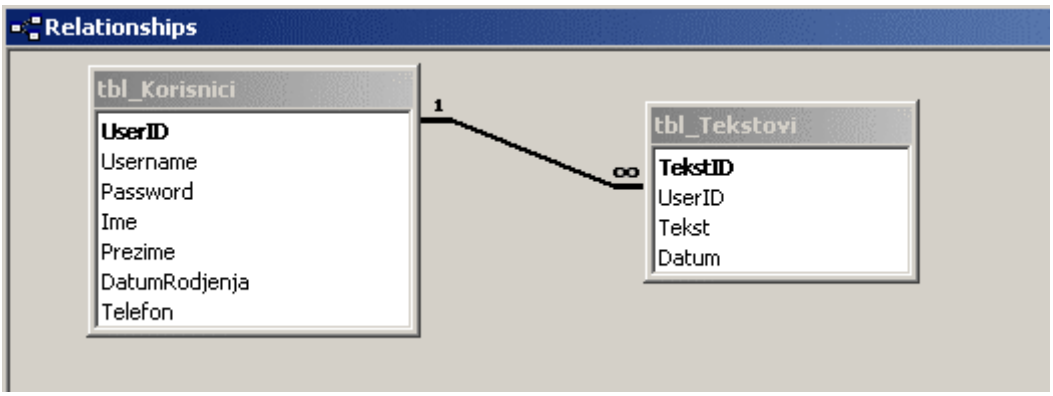
```
SELECT * FROM tbl_Tekstovi;
```

Dobivamo kao jedno od polja UserID ali to nam malo govori ko je autor teksta. Imamo brojeve, a ko će znati koji korisnik ima koji broj i slično tome. Lepše bi bilo da se kao jedno od polja pojavi npr. korisničko ime i odmah pročitamo da je autor tog teksta degojs, a tog drugog teksta – jc denton. Da vidimo..

```
SELECT tbl_Tekstovi.Tekst, tbl_Korisnici.Username
            FROM tbl_Tekstovi, tbl_Korisnici
            WHERE tbl_Tekstovi.UserID = tbl_Korisnici.UserID;
```

Da vidimo koji je postupak:

1. navedite sva polja koja trebate, ali u formatu tabela-tačka-polje.
    a. u našem primeru treba nam polje Tekst iz tabele tbl_Tekstovi pa pišemo tbl_Tekstovi.Tekst
    b. i treba nam polje Username iz tabele tbl_Korisnici pa pišemo tbl_Korisnici.Username
2. navedite sve tabele koje koristite u delu FROM
    a. pošto koristimo polja iz dve tabele pišemo FROM tbl_Tekstovi, tbl_Korisnici
3. navedite redom preko kojih polja su tabele povezane u delu WHERE.
    a. u našem slučaju imamo vezu preko polja UserID pa pišemo WHERE tbl_Tekstovi.UserID = tbl_Korisnici.UserID;
    Ova veza je lako vidljiva sa dijagrama koji smo videli još u prvom delu ovog uputstva kada smo kreirali bazu u Accessu. Da se prisetimo, dijagram izgleda ovako:




Dakle, valja imati negde pri ruci dijagram vaše baze i onda samo lepo u onom delu WHERE navedete kako su povezane tabele koje koristite u upitu sve redjajući veze tabelaA.poljeA = tabelaB.poljeB koristeći AND tabelaX.poljeX = tabelaY.poljeY AND ...

Eto, nije mnogo teško zar ne? Da vidimo samo kako da malo proširite upit sa još nekim uslovom:

```
SELECT tbl_Tekstovi.Tekst, tbl_Korisnici.Username, tbl_Tekstovi.Datum
            FROM tbl_Tekstovi, tbl_Korisnici
            WHERE tbl_Tekstovi.UserID = tbl_Korisnici.UserID
                  AND
                  tbl_Tekstovi.Datum > CDATE('21-NOV-2003 18:00');
```

Dakle, samo nakon što navedemo veze između tabela prosto dodamo još neki uslov na kraju: ovde dobivamo sve tekstove koji su napisani posle određenog datuma i vremena, a sve to lepo sa korisničkim imenom.

Eto i to bi bilo o JOIN. Jeste, ima još o istom, ali to već nije predmet ovog teksta. Samo da vas uputimo na to šta treba da potražite na Google i da pitate na EliteSecurity.org: OUTER JOIN, LEFT JOIN, RIGHT JOIN. Snaći ćete se već uz našu pomoć na forumu :-)

## *2.5. DELETE,*

Delete služi za brisanje zapisa je li. Primer:

```
DELETE FROM tbl_Korisnici WHERE UserID = 1;
```

Pre nego što se upustite u brisanje možete da vidite šta će biti obrisano sa:

```
SELECT * FROM tbl_Korisnici WHERE UserID = 1;
```

A kada se uverite da je to ono što želite obrisati, samo SELECT upit prepravite u DELETE izraz i gotova stvar.

## 2.6. UPDATE

Update, kako ime govori služi da promenimo podatke u poljima zapisa. Za primer, hajde da promenimo Ime i Prezime korisniku degojs. Do sad je bilo Dejan Gojsević, a sada da to promenimo u Zoran Zorić.

```
UPDATE tbl_Korisnici
      SET Ime = 'Zoran', Prezime = 'Zorić'
      WHERE UserID = 1;
```

Bitno je napomenuti da ne zaboravite da postavite uslov u WHERE jer će inače svi zapisi biti promenjeni !!!
Naime, ako probate

```
UPDATE tbl_Korisnici
      SET Ime = 'Zoran', Prezime = 'Zorić';
```

Desiće se prava katastrofa: svi korisnici u tabeli imaće ime Zoran i prezime Zorić. Dakle, ne zaboravite da tačno odredite koji zapis menjate.

## 2.7. INSERT.

Insert služi za ubacivanje novih zapisa u tabelu. Recimo da se novi korisnik registrovao i sad mi treba da ga dodamo u našu bazu, slično kako je EliteSecurity.org vas dodao u svoju bazu podataka kada ste se registrovali:

```
INSERT INTO tbl_Korisnici ( Username, [Password], Ime, Prezime, DatumRodjenja )
       VALUES ('Mirkom' , 'mmm1', 'Mirko', 'Mirić', #14-JUN-1985# );
```

NAPOMENA: polje “Password” morate ovde da stavire unutar uglatih zagrada kao u primeru gore jer inače Access neće znati da se radi o polju u tabeli već će to da protumači kao ključnu reč. Primetite da datum kod rada sa Access-om možemo da napišemo kao u primeru, #između taraba#.

Elem da mi vidimo kako izgleda ovaj gore izraz za dodavanje novih polja: pa lepo, navedete tabelu i polja, a zatim stavite ključnu reč VALUES i onda navedete nove vrednosti koje upisujete redom u polja kako ste ih prethodno naveli.

Primetite da nismo naveli polje UserID koje postoji u ovoj tabeli. Ovo nikako ne treba da navodite – ako se prisetite, ovo polje je AUTONUMBER što znači da će sama baza da mu dodeli vrednost.

Eto, i to bi bilo to – kraj našeg malog SQL uputstva. Za sva dodatna pitanja pravac www.EliteSecurity.org.

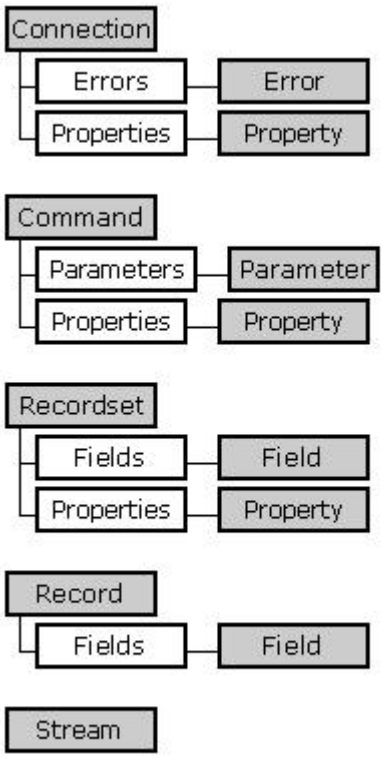
Pozdrav,

Dejan “degojs” Gojsević
VB & ASP forum @ www.EliteSecurity.org

Decembra, 2003.

# 3. Visual Basic 6 kod za rad sa bazom - ADO 2.5

## *ADO OBJEKTNI MODEL – KRATAK OPIS*



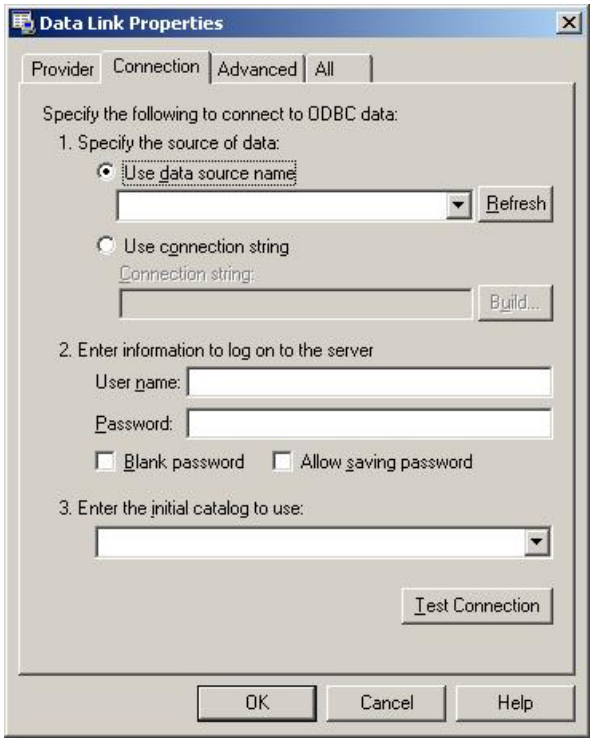

***Connection*** objekat je top level objekat u ADO hijerarhiji. Ovaj objekat predstavlja vezu ka data sourceu i preko njega ide sva komunikacija između VB aplikacije i data sourcea. Važniji metodi su: Open, Close, BeginTrans, CommitTrans i RollbackTrans.
***Command*** objekat predstavlja SQL izraz, stored proceduru ili neku drugu akciju nad podacima. *Parameters* kolekcija prihvata ulazne i izlazne parametre. Obično se koristi za parametrizovane kverije (query) i izvršavanje stored procedura koje vraćaju neki parametar.
***Recordset*** objekat predstavlja skup zapisa koji su rezultat nekog upita i kursor unutar njih. Sadrži kolekciju *Fields* koja sadrži *Field* objekte. *Field* objekat predstavlja jednu kolonu podataka unutar *Recordseta*.
***Error*** objekat sadrži informacije o grešci koju vraća OLE DB provider. Jedan SQL izraz može izazvati više grešaka, pa *Errors* kolekcija može sadržati više *Error* objekata koji su svi posledica istog SQL izraza.

## *3.1. Povezivanje sa bazom podataka - Connection String*

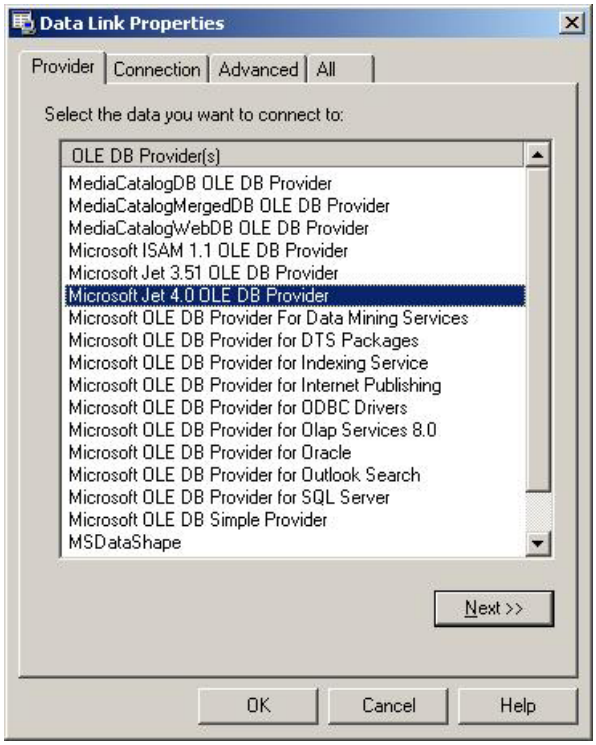
*Connection String* predstavlja informaciju koja se koristi za uspostavljanje veze sa *data source*om (izvorom podataka, bazom). Ovaj property sadrži niz parova *argument* = *vrednost*, odvojenih znakom ;.

Kako izgleda i kako se formira *Connection String*? Za ovo će poslužiti jedan primer koji je ***degojs*** par puta postovao na VB&ASP forumu :).

- Kreirajte fajl (iz Notepada, ili kako već ko voli) koji se zove, na primer, cs.udl (bitno je da ima ekstenziju UDL).
- Otvorite udl fajl iz Windows Explorera (double click, ili Open, ili ...)
- Trebalo bi da vam se pojavi jedan prozor kao na slici 3.1.1

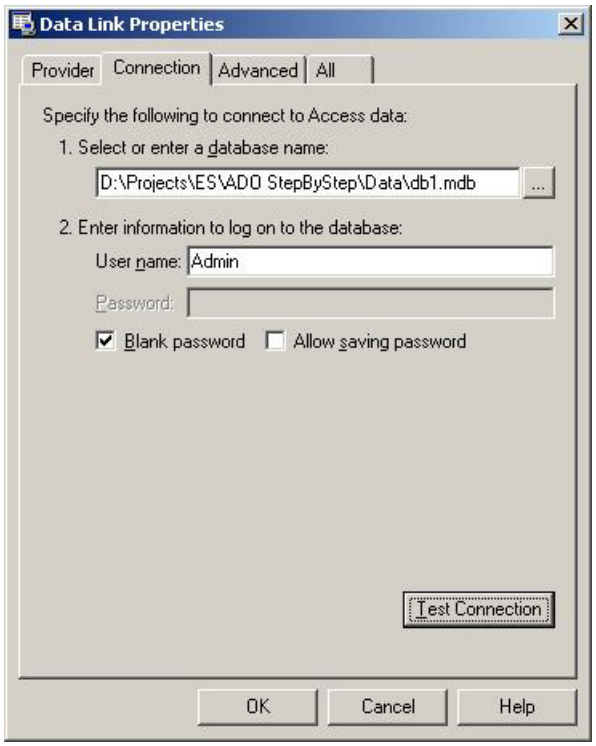



slika 3.1.1




slika 3.1.2

- Vratite se na prvi jezičak (Provider) – slika 3.1.2, i odaberite Providera. Za rad sa Accessom 2000 to će biti *Microsoft Jet 4.0 OLE DB Provider*
- Kliknite na na *Next*
- Na Connection jezičku odaberite putanju do baze (UserName ćemo ostaviti Admin, a password će biti prazan (slika 3.1.3)
- Sada smo sve podesili, i trebalo bi proveriti da li je sve kako treba. Kliknite na dugme test Connection.
- Ukoliko je sve u redu, pojaviće se poruka *Test connection suceeded* (slika 3.1.4)




slika 3.1.3


slika 3.1.4

- Kliknite na OK dugme
- Otvorite fajl u Notepadu
- Fajl bi trebao da izgleda ovako nekako (samo će se path do baze razlikovati):

```
[oledb]
; Everything after this line is an OLE DB initstring
Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=D:\Projects\ES\ADO StepByStep\Data\db1.mdb;Persist
Security Info=False
```

Naš *Connection String* izgleda ovako:

```
Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=D:\Projects\ES\ADO StepByStep\Data\db1.mdb;Persist
Security Info=False
```

OK. Sada imamo *Connection String*. Vreme je da ga iskoristimo.

## *3.2. Otvaranje konekcije, preuzimanje podataka, zatvaranje konekcije*

Uz ovu dokumentaciju ide i jedna VB aplikacija gde je sav kod za ono o čemu se ovde govori. Glavna forma sadrži menije i dugmiće za navigaciju kroz poglavlja ovog dokumenta.

Za početak da objasnimo kako se formira ConnectionString unutar aplikacije. U basMain modulu postoje tri konstante koje se koriste pri formiranju Connection Stringa:

```
'-- connection string constants
Private Const mcstrDSNStart As String =
"Provider=Microsoft.Jet.OLEDB.4.0;Data Source="
Private Const mcstrDSNEnd   As String = ";Persist Security
Info=False"
Private Const mcstrDbPath   As String = "\Data\db1.mdb"
```

Prve dve su fiksne, a treća (`mcstrDbPath`) predstavlja relativnu putanju do baze u odnosu na App.Path. ConnectionString se formira prilikom startovanja sample aplikacije u *Sub Main* (basMain modul)

```
Global gConnectionString    As String
Global gFSO                 As Scripting.FileSystemObject

...

gConnectionString = mcstrDSNStart &
                    gFSO.BuildPath(App.Path, mcstrDbPath) & _
                    mcstrDSNEnd
```

Na ovaj način smo dobili Connection String i koristićemo ga za sve dalje primere. Kod za ovaj primer se nalazi u frmChapter32 formi.

Kako ovo radi? Na formi treba selektovati iz koje tabele/kverija će se selektovati podaci i kliknuti na dugme. Nakon toga će se pojaviti podaci u text boxu sa desne strane.

1. za ovo nam trebaju sledeće promenljive

```
Private m_TableName             As String
...
Dim adoConn As ADODB.Connection ' konekcija
Dim adoRS   As ADODB.Recordset  ' rekordset sa podacima
Dim adoFld  As ADODB.Field      ' polje iz rekordseta
Dim sQry    As String           ' SQL upit
```

Promenljiva `m_TableName` predstavlja ime tabele/kverija odakle će se preuzimati podaci. Ova promenljiva se ažurira čim korisnik klikne na naziv tabele/kverija:

```
Private Sub optTable_Click(Index As Integer)
    ' u captionu se nalazi ime tabele/kverija
    m_TableName = optTable(Index).Caption
End Sub
```

2. kreiranje i otvaranje konekcije

```
    '-- kreiraj i otvori konekciju
    Set adoConn = New ADODB.Connection
    adoConn.Open gConnectionString
```

Konekcija se otvara metodom *Open* Connection objekta, a kao parametar se prosleđuje Connection String. Ako se Connection String ne prosledi kao parametar, onda se uzima vrednost ConnectionString propertya. Znači konekciju smo mogli i ovako da otvorimo:

```
    Set adoConn = New ADODB.Connection
    adoConn.ConnectionString = gConnectionString
    adoConn.Open
```

Pored Connection Stringa, Open metod prihvata i parametre *UserID*, *Password* i *Options*. *UserID* i *Password* se koriste da bi se prosledila informacija o korisniku koji se kači na bazu ukoliko je to potrebno (u ovom slučaju nije). *Options* parametar definiše način izvršavanja Open metoda – sinhrono (default) ili asinhrono. Ako se prosledi **`adAsyncConnect`** onda smo konektovani na bazu kada Connection objekat digne event *ConnectComplete*. U ovom slučaju bi Connection objekat trebalo deklarisati na nivou forme/klase i koristiti WithEvents u deklaraciji.

3. formiranje SQL upita za selekciju podataka

```
Public Const pcstrTableName     As String = "<TableName>"
Public Const pcstrTableQuery    As String = "SELECT * FROM " &
pcstrTableName
```

Ove konstante su definisane u basMain. Znači `pcstrTableQuery` izgleda ovako: `SELECT * FROM <TableName>`. Da bi selektovali podatke iz neke tabele/kverija, potrebno je `"<TableName>"` odnosno konstantu `pcstrTableName` zameniti nazivom željene tabele/kverija:

```
'-- formiraj upit
sQry = Replace(pcstrTableQuery, pcstrTableName, m_TableName)
```

4. kreiranje i otvaranje recordseta

```
    '-- kreiraj i otvori rekordset
    Set adoRS = New ADODB.Recordset
    adoRS.Open sQry, adoConn
```

Recordset objekat se otvara metodom Open. Prvi parametar je *Source* i u ovom slučaju je SQL upit. *Source* može biti Command objekat, poziv stored procedure, SQL upit, ime tabele ili naziv fajla koji sadrži persisted recordset (takav fajl se može dobiti pozivom adoRS.Save). Drugi parametar je aktivna konekcija. To može biti Connection objekat ili string koji predstavlja ConnectionString. Ostali parametri nisu navedeni pa se koriste default vrednosti. Ostali parametri su *CursorType*, *LockType*, *Options*. Ovi parametri zavise od toga šta ćemo raditi sa recordsetom. Za sada se neću baviti njima (u MSDNu ima sve lepo objašnjeno, a i mislim da će se kroz primere videti kad se koji koristi), a možda se kasnije vratim na ovo.

Recordset smo u ovom slučaju mogli otvoriti i ovako:

```
    adoRS.Open m_TableName, adoConn, , , adCmdTable
```

ili ovako:

```
    adoRS.CursorType = adOpenForwardOnly
    adoRS.LockType = adLockReadOnly
    adoRS.Open m_TableName, adoConn, , , adCmdTable
```

Znači *CursorType* i *LockType* parametri se mogu proslediti metodu Open, a mogu se setovati propertyji Recordset objekta.

O prolasku kroz recordset će biti reči u poglavlju 3.4.

## 3.3. Prikazivanje podataka u DataGrid kontroli

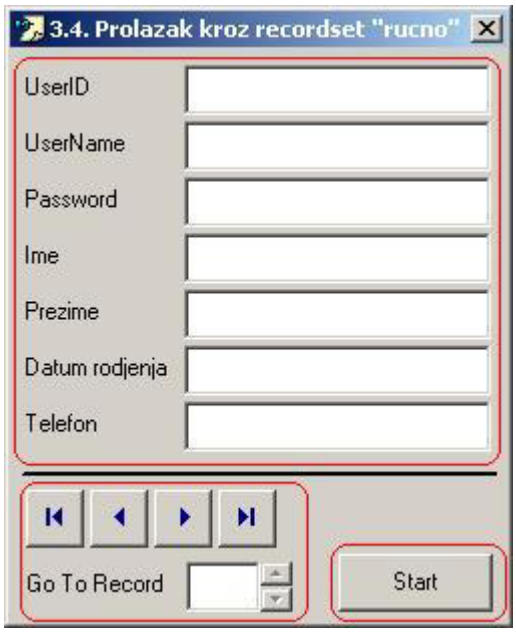
Kod ovog poglavlja se nalazi u formi frmChapter33. Ova forma sadrži jednu grupu option buttona koja služi za biranje tabele iz koje će se prikazivati podaci, DataGrid kontrolu u kojoj će se prikazati podaci i comand button koji otvara konekciju, preuzima podatke i prikazuje ih u gridu. Skoro sav kod je objašnjen u prethodnom poglavlju, a najbitniji deo je

```
'-- prikazi zapise
Set dgrData.DataSource = adoRS
```

DataSource property može biti *ADO Recordset*, kao i klasa ili kontrola koja je definisana kao DataSource (**DataSourceBehavior** property = **vbDataSource**).

## 3.4. Prolazak kroz recordset ručno

Kod ovog poglavlja se nalazi u formi frmChapter34.




Ova forma sadrži grupu textboxova u kojima će biti prikazani podaci iz tabele *tbl_Korisnici*, dugmiće za navigaciju i dugme *Start* (otvaranje konekcije i rekordseta i preuzimanje podataka).

Textboxovi u kojima se prikazuju podaci imaju podešen *Tag* property u design timeu na naziv polja iz recordseta čija će se vrednost prikazati u tom textboxu.

Metodi *Recordset* objekta koji su bitni za ovo poglavlje su: *Move*, *MoveFirst*, *MovePrevious*, *MoveNext* i *MoveLast*.

Propertyji *Recordset* objekta koji su nam ovde bitni su *BOF*, *EOF* i *AbsolutePosition*.

*BOF* i *EOF* su tipa Boolean. *BOF* je True ako je pozicija tekućeg zapisa u Recordset objektu pre prvog zapisa (indikator početka). *EOF* je True ako je pozicija tekućeg zapisa u Recordset objektu posle zadnjeg zapisa (indikator kraja). *EOF* i *BOF* su istovremeno True ako ne postoji tekući zapis (current record) tj. ako Recordset ne sadrži nijedan zapis. EOF i BOF služe kao granice za kretanje po Recordset objektu.

*AbsolutePosition* property omogućava čitanje/setovanje trenutne pozicije unutar *Recordset* objekta. Ovaj property vraća broj u intervalu od 1 do ukupnog broja zapisa u *Recordsetu*, ili jednu od sledeće tri konstante:

- adPosUnknown – ako je *Recordset* prazan, ili ako *provider* ne podržava property
- adPosBOF – ako je *BOF* property True
- adPosEOF – ako je *EOF* True

*MoveFirst*, *MoveLast*, *MovePrevious* i *MoveNext* metode neću pojašnjavati jer su nazivi metoda deskriptivni.

*Move* metod omogućava pomeranje za određeni broj zapisa napred ili nazad u odnosu na trenutnu poziciju ili u odnosu na bookmark (Bookmarke pogledati u MSDNu :)).

```
recordset.Move numrecords [,start]
```

`Numrecords` predstavlja broj zapisa za koliko treba izvrsiti pomeranje. Moye biti pozitivan ili negativan i od toga zavisi smer pomeranja. `start` parametar predstavlja zapis u odnosu na koga se vrši pomeranje. Ako se parametar start izostavi, podrazumeva se trenutna pozicija.

Kad se forma otvori, treba kliknuti na dugme Start. Tada se otvara konekcija, otvara se recordset i preuzimaju podaci. Zatim se proverava da li ima zapisa u recordsetu:

```
If Not (m_adoRS.EOF And m_adoRS.BOF) Then
```

Ako recordset nije prazan, podešavaju se parametri UpDown kontrole, koja će se koristiti za demonstraciju *Move* metoda:

```
        udPosition.Min = 1
        udPosition.Max = m_adoRS.RecordCount
```

Na kraju se prikazuje trenutni zapis i trenutna pozicija (`AbsolutePosition` property) u recordsetu:

```
        Call m_DisplayRecord
        txtPos.Text = m_adoRS.AbsolutePosition
```

`m_DisplayRecord` prikazuje trenutni zapis na formi. Zapis se prikazuje u nizu TextBoxova kojima je u design timeu u *Tag* property upisano ime polja koje će se prikazivati u tom textboxu. Kroz niz textboxova se prolazi *For Each ... Next* petljom i prikazuju se vrednosti iz recordseta.

```
    For Each oTextBox In txtField
        If Not IsNull(m_adoRS.Fields(oTextBox.Tag).Value) Then
            oTextBox.Text = m_adoRS.Fields(oTextBox.Tag).Value
        Else
            oTextBox.Text = ""
        End If
    Next
```

Ovo je moglo i drugačije da se implementira (imate u kodu i kako), ali ja radim ovako :).

E, stigosmo do pomeranja. Deo koda za kretanje kroz RS nalazi se u `cmdNavigation_Click`

```
    Select Case Index
        Case 0 ' move first
            m_adoRS.MoveFirst
        Case 1 ' move prev
            If Not m_adoRS.BOF Then m_adoRS.MovePrevious
            If m_adoRS.BOF And m_adoRS.RecordCount > 0 Then
                Beep
                'moved off the end so go back
                m_adoRS.MoveFirst
            End If
        Case 2 ' move next
            If Not m_adoRS.EOF Then m_adoRS.MoveNext
            If m_adoRS.EOF And m_adoRS.RecordCount > 0 Then
                Beep
                'moved off the end so go back
                m_adoRS.MoveLast
            End If
        Case 3  ' move last
            m_adoRS.MoveLast
    End Select
```

Prilikom pomeranja na sledeći ili prethodni zapis, potrebno je proveriti da li smo na kraju (EOF), odnosno početku (BOF).

U `udPosition_Change` može se videti kako se koristi *Move* metod. UpDown kontrola je napunjena brojevima od 1 do broja zapisa. Kad UpDown kontrola ima vrednost *n*, treba se pomeriti na *n*-ti zapis:

```
m_adoRS.Move udPosition.Value - 1, adBookmarkFirst
```

Gore navedeni kod pomera poziciju u recordsetu za n-1 u odnosu na prvi zapis. Ovo se moglo izvesti i ovako:

```
m_adoRS.AbsolutePosition = udPosition.Value
```

## *3.5. Izmena podataka (UPDATE, DELETE, INSERT)*

Podaci se mogu menjati izvršavanjem SQL izraza (UPDATE, DELETE, INSERT) ili u recordsetu metodama Delete, AddNew, Update i UpdateBatch. Pošto se ***degojs*** bavio privm načinom u drugom poglavlju ovog tutoriala, o tome neće biti reči ovde.

Kod koji se odnosi na ovo poglavlje nalazi se u formi frmChapter35. Ova forma liči na formu iz prethodnog poglavlja, s tim što ovde postoje dugmići za dodavanje novog zapisa, brisanje zapisa, izmenu podataka, snimanje izmena, odustajanje od izmena i osvežavanje podataka. Veći deo koda je isti kao u formi frmChapter34, pa će ovde biti objašnjen samo deo koji se odnosi na izmenu podataka.

### 3.5.1. DODAVANJE NOVOG ZAPISA (ADD)

```
'-- clear fields
Call m_ClearFields(txtField)
'-- otkljucaj kontrole
Call m_LockControls(txtField, False)
With m_adoRS
    If Not (.BOF And .EOF) Then
        '-- zapamti trenutnu poziciju da bi se vratili
        '-- ako korisnik odustane od novog zapisa
        m_Bookmark = .Bookmark
    End If
    .AddNew
    m_Mode = mdAdd
End With
'-- prikazi dugmice
m_ShowButtons cmdAction, "001010"
m_EnableButtons cmdNavigation, "0000"
```

Prvo treba obrisati vrednosti iz textboxova. Da su kontrole bile boundovane ovo bi se automatski desilo. Ako hoćete da radite sa boundovanim kontrolama, VB vam nudi VB Data Form Wizard prilikom dodavanja nove forme, tako da kad odgovorite na sva pitanja wizarda dobijate gotovu formu koja ima sličnu funkcionalnost kao ova naša, samo što su kontrole boundovane. Zatim, treba otključati polja da bi se u njih moglo nešto upisati. Sledeći korak je pamćenje trenutne pozicije u reordsetu, da bi mogli da se vratimo na tu poziciju ako se odustane od dodavanja novog zapisa. I sad ono najvažnije AddNew. Ovaj metod kreira novi zapis u recordsetu i postavlja ga za tekući. Nakon ovoga treba disableovati dugmiće za navigaciju, i prikazati samo Save i Cancel dugmiće. Sada treba uneti vrednosti za novi zapis,

kliknuti na Save da bi se zapis dodao u bazu, ili Cancel da bi se odustalo od dodavanja novog zapisa.

VAŽNO: Ne vrši se provera da li su podaci koji se unose validni (tip, dužina,...). Ukoliko se unese neispravan podataka baza će vratiti grešku.

### 3.5.2. IZMENA TRENUTNOG ZAPISA (EDIT)

```
'-- otkljucaj kontrole
Call m_LockControls(txtField, False)
m_Mode = mdEdit
'-- prikazi dugmice
m_ShowButtons cmdAction, "001010"
m_EnableButtons cmdNavigation, "0000"
```

Prvo se otključavaju polja da bi se mogle menjati vrednosti, prikazuju se odgovarajući dugmići i sad treba promeniti podatke i zapamtiti ih (Save) ili poništiti izmene (Cancel).

### 3.5.3. SNIMANJE PROMENA (SAVE)

```
'-- prepisi podatke iz textboxova u RS
Call m_UpdateData
'-- update
m_adoRS.UpdateBatch adAffectAll
'-- pozicioniraj se na novi/izmenjeni zapis
If m_Mode = mdAdd Then
    m_adoRS.MoveLast
    '-- new key is avaliable at this point
    txtField(0).Text = m_adoRS("UserID")
End If
m_Mode = mdNone
'-- prikazi dugmice
m_ShowButtons cmdAction, "110101"
m_EnableButtons cmdNavigation, "1111"
```

Za početak treba prepisati podatke iz textboxova u recordset (da su kontrole boundovane ovo se ne bi radilo). Zatim, ide poziv metoda *UpdateBatch.* Zavisno od parametra, ovaj metod će updateovati trenutni zapis (**adAffectCurrent**) ili sve zapise (**adAffectAll**). Ukoliko je dodat novi zapis, treba se pozicinirati na njega (dodaje se na kraj) i prikazati vrednost ključa. Pošto je ključ Autonumber, njegova vrednost se ne unosi, već baza vodi računa o tome. Nakon poziva metoda UpdateBatch informacija o IDu novog zapisa je dostupna. Na kraju treba prikazati dugmiće. Kad smo već ovde, dugmići su nizovi kontrola i enableuju se i prikazuju metodama `m_EnableButtons` i `m_ShowButtons`. Ove metode prihvataju stringove nula i jednica, pa ako je string npr. 101 to znači da će dugme sa indexom 0 biti

enableovano, dugme sa indexom 1 će biti disbleovano, a dugme sa indexom 2 će biti enableovano, ...

### 3.5.4. BRISANJE ZAPISA (DELETE)

```
With m_adoRS
    '-- brisi zapis
    .Delete
    '-- pomeri se na sledeci zapis
    .MoveNext
    If .EOF Then .MoveLast
End With
'-- update
m_adoRS.UpdateBatch adAffectAll
```

Metod *Delete* služi za brisanje zapisa. Pošto koristimo *LockType= adLockBatchOptimistic*, zapis će nakon poziva *Delete* metoda biti markiran za brisanje (nema brisanja iz baze), a tek nakon poziva UpdateBatch biće obrisan iz baze. Za više informacija pogledati LockType property u MSDNu.

Brisanje zapisa se moglo obaviti izvršavanjem sledećeg SQL izraza:

```
DELETE FROM tbl_Korisnici WHERE UserID=IDUsera
```

SQL izraz se izvršava pozivanjem *Execute* metode *Connection* objekta (pogledati u MSDNu)

Prilikom brisanja korisnika za kojeg postoje zapisi u tabeli tbl_Korisnici ćete dobiti grešku. Ovo se obično rešava čekiranjem opcije Cascade Delete Related Records na propertyjima veze izmedju dve tabele u samoj bazi. Otvorite bazu, pa Tools/Realtionships, selektujte vezu izmedju dve tabele, desni klik, pa Edit Relationship. Drugi način je da se pre brisanja korisnika, izvrši brisanje svih njegovih textova iz druge tabele.

```
m_adoConn.Execute "DELETE FROM tbl_Tekstovi WHERE UserID=IDUsera "
```

### 3.5.5. ODUSTAJANJE OD PROMENA (CANCEL)

*CancelUpdate* metod poništava promene koje su izvršene na tekućem zapisu ili na novom zapisu koji je dodat metodom AddNew. Ukoliko je poništeno dodavanje novog zapisa, potrebno je vratiti se na onaj zapis na kojem je kliknuto na Add.

```

```

```
            '-- cancel update
            Call m_LockControls(txtField, True)
            m_adoRS.CancelUpdate
            '-- pomeri se na zadnju poziciju ako je bilo AddNew
            If m_Bookmark > 0 Then
                m_adoRS.Bookmark = m_Bookmark
            Else
                m_adoRS.MoveFirst
            End If
            m_Mode = mdNone
            '-- prikazi dugmice
            m_ShowButtons cmdAction, "110101"
            m_EnableButtons cmdNavigation, "1111"
```

### 3.5.6. OSVEŽAVANJE PODATKA (REFRESH)

```
m_adoRS.Requery
```

*Requery* metod osvežava podatke ponovnim izvršavanjem upita čije podatke recordset sadrži.

## *3.6. Slanje podataka nazad u bazu (disconnected recordset)*

*Disconnected recordset* je rekordset koji je popunjen podacima iz *data sourcea* i nije nakačen na *data source* (disconnected), tj. ne postoji konekcija ka *data sourceu*. Sve promene koje se vrše su lokalne (ne vide se u bazi). Promene će se reflektovati na bazu ako se *recordset* objekat rekonektuje i ako se pozove *Update* ili *UpdateBatch* metod.

VB kod koji se odnosi na ovo poglavlje nalazi se u formi frmChapter36 i sličan je kodu iz forme frmChapter34. U čemu je razlika?

Kada se preuzmu podaci, *recordset* se diskonektuje i konekcija se zatvara:

```
    '-- diskonektuj RS
    Set m_adoRS.ActiveConnection = Nothing

    '-- zatvori konekciju
    m_adoConn.Close
    Set m_adoConn = Nothing
```

Pošto polja nisu boundovana, pre pomeranja sa tekućeg zapisa potrebno je upamtiti promene. To se radi u funkciji `m_UpdateData`.

```
'--prepisi podatke iz text boxova u RS
For Each oTextBox In txtField
  ' updateuj sve osim UserID (kljuc autonumber)
  If oTextBox.Tag <> "UserID" Then
    m_adoRS.Fields(oTextBox.Tag).Value = oTextBox.Text
  End If
Next
```

Ova funkcija ima opcioni parametar tipa *Boolean* koji govori da li treba rekonektovati *recordset*. Ova funkcija se poziva sa parametrom True (tj. recordset se rekonektuje) samo kad se klikine na dugme *Vrati podatke u bazu*. Znači, kreira se i otvara konekcija, a recordset se rekonektuje:

```
If bReconnectRS Then
    '-- kreiraj i otvori konekciju
    Set m_adoConn = New ADODB.Connection
    m_adoConn.Open gConnectionString

    '-- "rekonektuj" RS
    Set m_adoRS.ActiveConnection = m_adoConn
End If
```

Na kraju treba pozvati UpdateBatch metod:

```
'--updateuj podatke
If bReconnectRS Then m_adoRS.UpdateBatch
```

Nakon ovoga, promene koje su vršene nad podacima videće se i u bazi.

Eto toliko. Ako naidjete na bugove, prijavite ih i srećno programiranje ☺

Pozdrav,
Željko Mladenović, aka mladenovicz