

UNIVERZA V LJUBLJANI  
FAKULTETA ZA MATEMATIKO IN FIZIKO

Matematika – praktična matematika (VSŠ)

Renata Sedej

IZDELAVA PODATKOVNE BAZE V PROGRAMU MS ACCESS

Diplomska naloga

Ljubljana, 2006

## KAZALO

<b>1</b>	<b>UVOD</b> .....	<b>5</b>
<b>2</b>	<b>OSNOVE PODATKOVNIH BAZ</b> .....	<b>6</b>
2.1	DEFINICIJA PODATKOVNE BAZE.....	6
2.2	RELACIJSKA PODATKOVNA BAZA .....	6
2.2.1	<i>Osnovni pojmi</i> .....	7
2.2.2	<i>Relacije med podatki</i> .....	8
<b>3</b>	<b>OPIS PROBLEMA</b> .....	<b>10</b>
<b>4</b>	<b>ZAKAJ MS ACCESS</b> .....	<b>12</b>
<b>5</b>	<b>REŠITEV PROBLEMA</b> .....	<b>13</b>
5.1	NAČRTOVANJE BAZE .....	13
5.1.1	<i>Želje uporabnikov</i> .....	13
5.1.2	<i>Razporeditev podatkov med tabele</i> .....	14
5.1.3	<i>Določanje polj v tabelah</i> .....	15
5.1.4	<i>Določanje ključev</i> .....	16
5.1.5	<i>Relacije med tabelami</i> .....	17
5.1.6	<i>Pregled načrta skupaj z uporabnikom</i> .....	22
5.2	IZDELAVA TABEL.....	28
5.2.1	<i>Izdelava tabel v pogledu načrta</i> .....	29
5.2.2	<i>Izdelava tabel s čarovnikom</i> .....	30
5.2.3	<i>Izdelava tabel z vnašanjem podatkov</i> .....	31
5.2.4	<i>Vzpostavljanje relacij med tabelami</i> .....	37
5.2.4.1	Referenčna integriteta .....	39
5.2.4.2	Vrsta stika .....	44
5.2.4.3	Vzpostavljanje relacij.....	46
5.2.4.4	Vzpostavljanje relacij v bazi PisaMS06.....	49
5.2.5	<i>Vnos podatkov v tabele</i> .....	52
5.3	IZDELAVA POIZVEDB .....	59
5.3.1	<i>Enostavne poizvedbe</i> .....	59
5.3.2	<i>Sestavljene poizvedbe</i> .....	61
5.4	IZDELAVA OBRAZCEV .....	69
5.4.1	<i>Preprosti obrazci</i> .....	69
5.4.2	<i>Kompleksni obrazci</i> .....	72
5.5	IZDELAVA POROČIL .....	76
5.6	IZDELAVA STIKALNE PLOŠČE.....	81
<b>6</b>	<b>ZAKLJUČEK</b> .....	<b>84</b>
<b>7</b>	<b>LITERATURA</b> .....	<b>85</b>

## Program dela

V diplomski nalogi na praktičnem primeru organizacije kontaktnih podatkov raziskave PISA prikažite, kako poteka izdelava podatkovne baze s programom Microsoft Access. Ob tem osnovno razložite tudi pojme kot so relacijska baza, referenčna integriteta, tipi relacij med tabelami, poročila in podobno.

mentor

mag. Matija Lokar

## **Povzetek**

*Diplomska naloga opisuje izdelavo podatkovne baze v programu MS Access. Izdelava je prilagojena potrebam organizacije kontaktnih podatkov raziskave PISA.*

*Opisane so osnovne podatkovnih baz. Tretje poglavje vsebuje podrobnejši opis, kaj je povod za izdelavo diplomske naloge in kaj bomo z njo dosegli. Četrto poglavje pojasnjuje zakaj smo se odločili za izdelavo podatkovne baze prav s programom MS Access.*

*Peto poglavje vsebuje podroben opis načrtovanja tabel kot tudi njihovo izdelavo. Ob izdelavi so opisani tudi potrebni pojmi, kot so referenčna integriteta, notranji stik, zunanji stik, različne relacije med tabelami itd. V nadaljevanju poglavja je opisana izdelava poizvedb, obrazcev in poročil, ki temeljijo na predhodno izdelanih tabelah in vnešenih podatkih.*

**Math. Subj. Class. (2000):** 68P15

**Computing Review Class. System:** E.5

**Ključne besede:** relacijska podatkovna baza, MS Access

**Keywords:** relational database, MS Access

## 1 UVOD

Količina podatkov od nas zahteva, da si številne podatke skrbno uredimo in ohranjamo v obliki, ki omogoča enostaven in jasen pregled teh podatkov. Uporaba informacijske tehnologije zahteva podatke v elektronski obliki, kar omogoča hitro izmenjavo podatkov med različnimi uporabniki. Potrebno je skrbeti tudi za varnost podatkov, ki so uporabnikom vedno lažje dostopni preko svetovnega spleta. Vedno večje potrebe k sreči povzročajo tudi vedno večjo ponudbo številnih rešitev organiziranja, hranjenja in varovanja podatkov. Temu so namenjeni številni sistemi za upravljanje podatkovnih baz.

Majhen košček k organizaciji podatkov sem prispevala tudi jaz v diplomski nalogi s tem, da sem organizirala podatke, ki sem jih pri delu tudi uporabljala. Želela sem rešiti konkreten problem, zato teorija, ki stoji za delom, ni podrobneje opisana. Trudila sem se upoštevati teorijo toliko, da bi bila rešitev po moji presoji optimalna. Glede na moje pomanjkanje splošnega znanja o podatkovnih bazah verjamem, da bi se dalo nekatere stvari narediti bolje.

V času študija podatkovnih baz nismo podrobneje obravnavali. Spoznali smo le nekaj osnovno prosto dostopne podatkovne baze MySQL. Izdelava podatkovne baze v MS Accessu je od mene zahtevala samostojno spoznavanje tematike in njeno uporabo. Za končen izdelek je bilo potrebno samostojno delo in prebiranje različne literature. Bralec naj upošteva, da so za izdelavo podatkovne baze »PisaMS06« v tej nalogi uporabljene osnovne funkcije programa Access in bi se dalo s profesionalnim znanjem narediti veliko več. V nalogi dajem poudarek uporabi določenih orodij programa Access in ne temu, kako delujejo.

K diplomski nalogi sem torej pristopila s praktičnega stališča in ne teoretičnega. Moj glavni cilj je izdelati podatkovno bazo, ki bo zadovoljila potrebe organizacije kontaktnih podatkov raziskave PISA, pri kateri sem sodelovala kot skrbnik podatkov raziskave. Pravih podatkov v nalogi ne omenjam zaradi varovanja osebnih podatkov. V diplomsko nalogo sem vključila veliko slik in s tem poskušala bralcu približati potek izdelave podatkovne baze. Samo podatkovno bazo si je možno ogledati na priloženi zgoščenki. V njej ni pravih podatkov zopet zaradi njihovega varovanja. Dovoljujem njeno prosto uporabo in prilagajanje potrebam uporabnika.

## 2 OSNOVE PODATKOVNIH BAZ

### 2.1 Definicija podatkovne baze

V literaturi najdemo številne definicije podatkovnih baz. Navedimo eno izmed njih, ki jo najdemo v [1]:

*Podatkovna baza - podatkovna osnova - je zbirka med seboj pomensko povezanih podatkov, ki so shranjeni v računalniškem sistemu, dostop do njih je centraliziran in omogočen s pomočjo sistema za upravljanje podatkovnih baz.*

Gre torej za to, da nek proces iz realnega sveta preslikamo v programsko obliko in podatke o njem hranimo v elektronski obliki na enem mestu. V našem primeru bomo beležili podatke o izvajanju projekta PISA. Kaj ta projekt je, bomo razložili v nadaljevanju. Za to, da je upravljanje s podatki čimbolj enostavno in pregledno, poskrbi tako imenovani sistem za upravljanje podatkovnih baz (SUPB). V našem primeru bo to program MS Access.

### 2.2 Relacijska podatkovna baza

Relacijska podatkovna baza se uporabniku kaže kot množica tabel, sestavljena iz vrstic (z imenom zapisi) in stolpcev (z imenom atributi ali polja). Vsaka tabela je sestavljena iz čelne vrstice in podatkovnih vrstic. Čelna vrstica vsebuje imena posameznih stolpcev v tabeli.

Vsebina tabel je uporabniku dostopna s pomočjo poizvedovalnih jezikov. Poizvedovalni jezik, ki je doživel standardizacijo v svetovnem merilu, je SQL (Structured Query Language), ki so ga razvili v podjetju IBM. Po celem svetu se je razširil predvsem zaradi učinkovitosti in preproste uporabe v sistemih za upravljanje relacijskih podatkovnih baz (relational database management system - RDBMS). Odobrila ga je tako organizacija ANSI (American National Standards Institute), kakor tudi ISO (International Standards Organization) in ga prvič zapisala v standardu SQL-89 leta 1989. SQL ni le jezik, v katerem pišemo poizvedbe (query) po določenih podatkih v bazi, ampak vsebuje tudi številne druge ukaze in funkcije. Delimo ga na dva dela:

- Jezik za rokovanje s podatki (DML - Data Manipulation Language) je podmnožica SQL-a, ki omogoča uporabnikom tako izvajanje povpraševanj po podatkih z določenimi lastnostmi kot tudi dodajanje, brisanje ter spreminjanje zapisov v podatkovni bazi. Poizvedovanje po podatkih izvajamo preko ukaza SELECT. Za spreminjanje podatkov uporabljamo ukaze DELETE (za brisanje podatkov), INSERT (za vstavljanje podatkov) in UPDATE (za posodabljanje podatkov).
- Jezik za definiranje podatkov (DDL - Data Definition Language) je prav tako podmnožica SQL-a, ki pa podpira kreiranje, brisanje in spreminjanje definicij tabel in pogledov na zapise. DDL obsega tudi ukaze za specifikacijo dostopnih pravic do tabel in pogledov. Nekateri najpomembnejši ukazi tega dela jezika so CREATE TABLE (ustvari tabelo), ALTER TABLE (spremeni tabelo), DROP TABLE (izbriše tabelo), CREATE INDEX (ustvari indeks) in DROP INDEX (izbriše indeks).

Določimo lahko tudi omejitve, ki jim morajo podatki zadoščati ob vnosu v bazo in podobno. Prav tako s pomočjo ukazov jezika SQL lahko določimo pristopna dovoljenja, torej povemo, kateri uporabniki lahko dostopajo do baze, spreminjajo in brišejo določene podatke, spreminjajo strukturo baze in podobno.

Čeprav je jezik SQL standardiziran, pa večina sistemov za upravljanje podatkovnih baz podpira svojo različico jezika SQL. Ta je sicer v skladu s standardom, a ga pogosto še nekoliko razširja. V našem primeru MS Access 2003 vsebuje podatkovni stroj (database engine) za upravljanje s podatki MS JetSQL, ki razširja standard ANSI SQL-92.

### 2.2.1 Osnovni pojmi

V tem razdelku bomo predstavili pojme oziroma izraze, ki jih bomo uporabljali pri delu s podatkovnimi bazami. Za lažje razumevanje si izmislimo bazo, sestavljeno iz treh tabel. Predstavlja naj preprosto evidenco zaposlenih v nekem podjetju. Prva tabela naj se imenuje ZAPOSLENI in naj vsebuje osnovne informacije o zaposlenih.

IDZaposlenega	ImePriimek	IDOddelka	Stopnja
MK1	Miha Kranjc	PO	IV
MH1	Maja Hrust	KO	VII
LS1	Lidija Svet	RO	V
MK2	Mitja Kern	NO	IV

Tabela 1 Tabela ZAPOSLENI

S pomočjo te tabele pojasnimo osnovne pojme kot so entiteta, tabela, atribut, zapis, celica, čelna vrstica, podatkovne vrstice itd.

**Entiteta** je stvar ali dogodek, ki ga opazujemo (npr. zaposleni). Značilnosti entitete opišemo z atributi (npr. *ImePriimek*). Podatke o entitetah shranjujemo v tabelah. Vsaki entiteti (v našem primeru je ta entiteta vsi zaposleni v podjetju) pripada ena tabela. Vsaka vrstica v tabeli (zapis) ustreza enemu primerku entitete (enemu zaposlenemu). Vsak stolpec v tabeli pomeni en atribut – lastnost entitete (na primer stopnja izobrazbe v stolpcu Stopnja). Vsak stolpec ima prirejen podatkovni tip (besedilo, število itd). Med entitetami veljajo določeni odnosi, ki jim pravimo relacije.

Zgoraj je prikazana dvodimenzionalna tabela (tabela 1), ki predstavlja entiteto ZAPOSLENI in je osnovna enota relacijske podatkovne baze. Dvodimenzionalna tabela je sestavljena iz vrstic (zapis) in stolpcev (polje, atribut). Stičišču vrstice in stolpca pravimo celica. Na sliki 1 je z rumeno barvo prikazan stolpec, z zeleno vrstica in z modro njuno stičišče - celica. O polju govorimo tudi pri posameznem zapisu. Tako je 'Miha Kranjc' vrednost polja *ImePriimek* prvega zapisa.

IDZaposlenega	ImePriimek	IDOddelka	Stopnja
MK1	Miha Kranjc	PO	IV
MH1	Maja Hrust	KO	VII
LS1	Lidija Svet	RO	V
MK2	Mitja Kern	NO	IV

Slika 1 Prikaz vrstice (zeleno), stolpca (rumeno) in celice (modro)

Čelna vrstica (rumena vrstica na sliki 2) vsebuje informacije o posameznih poljih tabele. Naša tabela vsebuje štiri polja in sicer *IDZaposlenega*, *ImePriimek*, *IDOddelka* in *Stopnja* (stopnja izobrazbe). *IDZaposlenega* v prvem polju je primarni ključ tabele.

IDZaposlenega	ImePriimek	IDOddelka	Stopnja
MK1	Miha Kranjc	PO	IV
MH1	Maja Hrust	KO	VII
LS1	Lidija Svet	RO	V
MK2	Mitja Kern	NO	IV

Slika 2 Čelna vrstica (rumeno) in podatkovne vrstice (zeleno)

**Primarni ključ** je eno ali več polj (stolpcev) v tabeli, katerih vrednosti enolično določajo vsak zapis v tabeli. Vsak zapis mora imeti primarni ključ z določeno vrednostjo, torej primarni ključ ne sme imeti ničelne (null) vrednosti (vrednosti, s katero označujemo manjkajoče oz. neznanе podatke). Primarni ključ je lahko zaporedno število, ni pa nujno. Pomembno je le, da je enoličen. To pomeni, da polje s primarnim ključem ne vsebuje dveh popolnoma enakih podatkov. S primarnim ključem torej dosežemo, da se dva zapisa razlikujeta vsaj v vrednosti primarnega ključa. To je dovolj, da ju SUPB obravnava kot dva različna zapisa. Primarni ključ uporabljamo tudi za povezovanje (relacije) s tujimi ključi v drugih tabelah. Več o relacijah in o tem kaj je tuji ključ, bomo opisali v razdelku 2.2.2.

Podatkovne vrstice (zelenе vrstice na sliki 2) - zapisi predstavljajo posamezne primerke entitet in vsebujejo konkretne vrednosti posameznih polj. Pri nas vrstica oz. zapis predstavlja zaposlenega posameznika v podjetju (slika 2). Primarni ključ vsakega zapisa predstavlja enolično oznako zaposlenega posameznika v podjetju.

IDOddelka	Naziv	Lokacija
PO	Prodajni oddelek	S1N2
KO	Kadrovski oddelek	S10N1
RO	Računovodski oddelek	S3N3
NO	Nabavni oddelek	S14N2

Tabela 2 Tabela ODDELEK

Drugo tabelo ustvarimo, da v njej hranimo podatke o posameznih oddelkih v podjetju. Tabela ODDELEK prav tako sestavljajo tri polja in sicer *IDOddelka*, *Naziv* in *Lokacija*. Polje *IDOddelka* je primarni ključ. Polje *Naziv* vsebuje opis oddelka. V polju *Lokacija* hranimo informacijo v kateri sobi in v katerem nadstropju podjetja se oddelek nahaja. Predpostavimo, da ima vsak oddelek podjetja le eno sobo.

IDZaposlenega	Naslov	Pošta	Kraj	TelDoma
MK1	Črnuška 1	1000	Ljubljana	01/123-45-67
MH1	Prešernova 7	2000	Maribor	02/765-41-23
LS1	Hubadova 3	1000	Ljubljana	01/567-24-31
MK2	Gregorčičeva 15	3000	Celje	03/432-15-67

Tabela 3 Tabela NASLOVI

Tretjo tabelo NASLOVI ustvarimo za hranjenje osebnih podatkov o zaposlenih. Damo ji enak primarni ključ kot tabeli ZAPOSLENI. Zakaj je ključ enak, bomo pojasnili v naslednjem razdelku.

## 2.2.2 Relacije med podatki

Podatke v podatkovni bazi razdelimo na posamezne tabele. Če želimo kasneje informacije iz več tabel združiti v poizvedbi, obrazcu, poročilu ali strani za dostop do podatkov, definiramo relacije med tabelami. Relacije v splošnem opisujejo odnose med podatki v tabelah.



Relacija v podatkovni bazi povezuje zapise dveh tabel, ki imata ujemajoče se podatke v poljih s ključem. Prva tabela v relaciji, ki ji pravimo tudi primarna tabela, vsebuje primarni ključ. Druga tabela, s katero povezujemo prvo tabelo, mora vsebovati tuji ključ. Kaj je to tuji ključ, bomo pojasnili v nadaljevanju. Običajno se primarni in tuji ključ nahajata v poljih z enakima imenoma v obeh tabelah, ni pa to nujno. Morata pa povezani polji biti istega podatkovnega tipa, da njune podatke lahko primerjamo. Pri primerjanju podatkov v programu MS Access upoštevajmo, da je podatkovni tip Samoštevilo (kaj to je, bomo pojasnili v nadaljevanju) v bistvu isti podatkovni tip kot Število, torej je lahko primarni ključ npr. tipa Samoštevilo, tuji ključ pa tipa Število. Pri primerjanju števil (in Samoštevil) moramo upoštevati še dodatni pogoj. Lastnost VelikostPolja obeh polj mora biti enaka. Polji tipa Samoštevilo in Število lahko na primer primerjamo, če je lastnost VelikostPolja obeh polj Dolgo celo število.

Kaj je torej tuji ključ? Tuji ključ je eno ali več polj (stolpcev) v tabeli, ki se sklicujejo na polje ali več polj primarnega ključa v primarni tabeli. Tuji ključ kaže na relacije med tabelami. Če pogledamo tabelo ZAPOSLENI (tabela 1) vidimo, da vsebuje polje z imenom *IDOddelka*. To je tuji ključ, ki nam omogoči povezavo s tabelo ODDELEK (tabela 2), v kateri polje *IDOddelka* predstavlja primarni ključ. S pomočjo tujega ključa *IDOddelka* v tabeli ZAPOSLENI bomo iz tabele ODDELEK za posameznega zaposlenega pridobili želene podatke o oddelku, na katerem dela.

Kasneje pri izdelavi baze se bomo z relacijami srečali bolj praktično in jih tudi podrobneje opisali.

### 3 OPIS PROBLEMA

Na delovni praksi v javnem raziskovalnem zavodu Pedagoški inštitut sem se srečala z izvedbo projekta PISA. PISA (Programme for International Student Assessment) je mednarodna raziskava o bralni, matematični in naravoslovni pismenosti. Izvaja se pod okriljem Organizacije za ekonomsko sodelovanje in razvoj (OECD) in poteka v triletnih ciklih. V prvem ciklu, ki je bil izveden v letu 2000, so sodelovale države članice OECD in le nekaj držav partnerk (nečlanice OECD). V drugem ciklu je poleg držav članic sodelovalo še 11 držav partnerk. V tretjem ciklu zajema podatkov PISA, ki se je začel izvajati v letu 2004, pa poleg držav članic OECD sodeluje še 27 držav partnerk, med njimi prvič tudi Slovenija. Glavnina podatkov tretjega cikla se zajema v letu 2006. V raziskavo so zajete 15-letne učenke, učenci, dijakinje in dijaki, ne glede na vrsto šole, ki jo obiskujejo. Namen raziskave PISA je zajeti podatke o kompetencah učencev, ki so pomembne tako za posameznika kot za celotno družbo. PISA meri znanje in veščine, ki so potrebne v življenju posameznika in družbe in ni posebej usmerjena na merjenje rezultatov šolskih kurikulov. To na nek način omejuje možnosti raziskovanja povezav med razlikami v dosežkih učencev in razlikami v načrtovanih in izvedenih kurikuli v posameznih državah ali med državami. Hkrati pa z zajemom populacije 15-letnih učencev ne glede na stopnjo šolanja omogoča učinkovito merjenje rezultatov šolskih sistemov in primerjavo teh rezultatov med državami. Izvajanje projekta poteka po strogih mednarodnih standardih in pravilih, kar kasneje omogoča primerjanje zbranih podatkov z ostalimi sodelujočimi državami. Mednarodna baza podatkov bo za tretji cikel dokončno pripravljena predvidoma decembra 2007.

Izvedba projekta v prvi fazi zahteva veliko organizacije in kontaktov s šolami, na katerih se raziskava izvede. V času mojega prihoda na Pedagoški inštitut so bili kontaktni podatki večinoma že zbrani. Shranjeni so bili v Excelovih datotekah na več računalnikih, ker na projektu dela več zaposlenih. Zaradi tega je bilo vzdrževanje podatkov precej zapleteno in nepregledno. Včasih so bili podatki podvojeni, včasih pa so se pri komu tudi izgubili. Bile so tudi druge pomanjkljivosti v sami organizaciji podatkov. Tako na primer pri podatkih o izvedbah posameznih testiranj ni bilo vpogleda v to, ali je v preglednici ravnateljev naveden ravnatelj te šole ali ne. Skratka podatki v omenjenih datotekah med seboj niso bili popolnoma usklajeni, niso bili povezani, nekaj podatkov se je podvajalo, vnašalo se jih na več mestih itd. Torej se je s to strukturo kršilo večino pravil organiziranega hranjenja podatkov. Kljub pomanjkljivostim nam je uspelo ta del projekta uspešno izvesti. Vendar je kmalu postalo očitno, da je potrebno obstoječi pristop k organizaciji podatkov spremeniti. To me je spodbudilo k izdelavi te diplomske naloge, ki se ukvarja predvsem z organizacijo in hranjenjem kontaktnih podatkov projekta PISA.

Potrebno je pojasniti nekaj izrazov, ki jih bomo uporabljali ves čas reševanja problema.

- Mednarodni center je Mednarodno združenje organizacij (Consortium) iz področja izvajanja nacionalnih in mednarodnih raziskav znanja, ki ga je na podlagi razpisa izbral OECD v letu 1997.
- Nacionalni center je nacionalni center raziskave PISA s sedežem na Pedagoškem inštitutu.
- Uporabnik je član osebja Pedagoškega inštituta, ki dela pri projektu PISA.
- Izvedba je 3-urno testiranje ene vzorčne enote. Vzorčna enota je 25 naključno izbranih dijakov iz posameznega izobraževalnega programa določene šole. Velikost vzorčne enote je odvisna od dogovora z mednarodnim centrom in od števila 15-letnih dijakov v posameznem izobraževalnem programu.
- Koordinator je profesor ali ravnatelj, ki organizira izvedbo na posamezni šoli.
- Izvajalec je s strani nacionalnega centra določena oseba, ki izpelje izvedbo na šoli. Dijakom daje navodila, jim prinese testne pole in rešene teste dostavi nazaj v nacionalni center.

V tem ciklu raziskave so v Sloveniji v raziskavo vključene vse gimnazije, ter srednje in nekatere osnovne šole. Za vsak učni program na šoli se izvede po ena izvedba. Pri označevanju posameznih izvedb se moramo prilagoditi mednarodno določenim identifikacijskim številkam (StrID in SchID), ki se v vsakem ciklu raziskave spremenijo.

Bralec bo verjetno imel določene pomisleke in težave pri razumevanju naših odločitev, ker ne pozna praktične narave in pravil projekta. Zato bom skušala sproti pojasnjevati dejanske potrebe in okoliščine, ki so nas vodile k določenim odločitvam. Izraz »naš problem« bo v nadaljevanju označeval izdelavo baze v MS Accessu in čimbolj organizirano vodenje podatkov.

in fiziko  
matematiko za  
Fakulteta

#### 4 ZAKAJ MS ACCESS

Microsoft Access je eden izmed sistemov za upravljanje relacijskih podatkovnih baz. Deluje v operacijskem sistemu Microsoft Windows in je del paketa Microsoft Office.

Sistemi za upravljanje podatkovnih baz so namenjeni zbiranju, organiziranju in vzdrževanju podatkov, ki jih lahko na različne načine obdelujemo – pregledujemo, spreminjamo, preračunavamo in grafično predstavljamo na monitorju, papirju ali na svetovnem spletu.

V našem primeru smo se za MS Access odločili zaradi njegove razširjenosti in preproste uporabe. Od Pedagoškega inštituta ni zahteval nobenih dodatnih vlaganj v programsko opremo, saj imajo vsi bodoči uporabniki naše baze program MS Access že nameščen na svojih računalnikih. Prav tako ne zahteva bistvenega dodatnega izobraževanja uporabnikov, ker bomo s pomočjo uporabe obrazcev in vnaprej definiranih poizvedb in izpisov, uporabo kar se da poenostavili. Hkrati tudi popolnoma zadovoljuje naše potrebe, saj bomo informacije lahko urejali v eni datoteki na enem mestu. Znotraj datoteke bomo uporabljali:

- tabele, kjer bomo hranili podatke;
- poizvedbe za iskanje in pridobivanje želenih podatkov;
- obrazce (zaslonske maske oz. forme) za ogled, dodajanje in posodabljanje podatkov v tabelah;
- poročila za analiziranje in tiskanje podatkov v določeni postavitvi;
- strani za dostop do podatkov, za gledanje, posodabljanje in analiziranje baze podatkov.

Vsi podatki iz do sedaj ločenih Excelovih datotek bodo dostopni na enem mestu, kar bo odpravilo težave z nekonsistentnostjo podatkov ter potrebo po nenehnem usklajevanju teh datotek pri vseh uporabnikih. Tudi količina podatkov in število uporabnikov ne presega zmoglosti programa Access. Podatkovno bazo bo uporabljalo manj kot 5 uporabnikov. Po oceni bo obsegala manj kot 10 tabel, od katerih nobena ne bo imela več kot 500 zapisov.

## 5 REŠITEV PROBLEMA

### 5.1 Načrtovanje baze

Načrtovanje baze razdelimo v več točk, kjer po korakih oblikujemo bodočo bazo podatkov. Dejansko točke med seboj niso strogo ločene, ampak se ponekod prepletajo.

#### 5.1.1 Želje uporabnikov

Pri načrtovanju baze upoštevamo želje uporabnikov in strukturo že obstoječih podatkov. V našem primeru uporabniki pri telefonskem komuniciranju s koordinatorji ali izvajalci potrebujejo hitre odgovore na številna vprašanja:

- Kateri izvajalec ima danes izvedbo in na kateri šoli?
- Katere izvedbe bo (ali je) izvajalec izvedel v nekem časovnem obdobju?
- Katere šole so v določeni območni enoti?
- Kateri izvajalci na določen datum nimajo načrtovane nobene izvedbe?
- Ali ima določen izvajalec danes izvedbo in katero?
- Kdo je koordinator na določeni šoli?
- Na kateri šoli je določena oseba koordinator?
- Koliko izvedb vodi določen koordinator?
- Kateri izvajalec izvaja določeno izvedbo?
- Na kateri šoli potekajo izvedbe vzporedno?
- Itd.

Uporabniki nekatere informacije želijo natisniti, kar jim lahko omogočimo z ustreznimi poročili. Pri pošiljanju pošte potrebujejo naslove izbranih oseb. Prav tako si želijo podatke vnašati preko vnosnih mask, ki bi omogočale hiter in preprost vnos podatkov.

Glede na veliko število funkcij, ki jih bomo morali izdelati, bo smiselno oblikovati vmesnik, na katerem bodo gumbi za dostop do vseh potrebnih funkcij. S tem bo uporabnikom na enem mestu omogočen dostop do vseh funkcij. Z besedo funkcija mislimo na posamezno nalogo, ki jo bo morala baza omogočiti. Ena od funkcij je na primer, da se ob kliku na gumb izdelava poročila z naslovi šol.

Naštejmo poročila in obrazce, ki jih bo potrebno izdelati, da si bodo uporabniki lahko odgovorili na zgoraj zastavljena vprašanja:

#### POROČILA:

- naslovi koordinatorjev v obliki primerni za tiskanje na nalepke,
- izpis izvedb, razvrščenih po različnih parametrih: po izvajalcih, po časovnem obdobju in po koordinatorjih,
- podatki o šolah,
- podatki o izvajalcih.

#### OBRAZCI:

- obrazec za vnos izvajalcev,
- obrazec za vnos šol,
- obrazec za vnos izvedb,
- obrazec za vnos koordinatorjev in ravnateljev.

Končno število in obliko obrazcev ter poročil bomo določili pri sami izdelavi le teh. Takrat bomo iskali tudi konkretne odgovore na posamezna vprašanja.

Kot smo že omenili, so podatki trenutno vodeni v več Excelovih datotekah. V prvi datoteki so shranjeni podatki o izvedbah. Zabeležene so informacije o šoli, mednarodna oznaka posamezne izvedbe (zaradi različnih programov je na eni šoli več izvedb), število vzorčenih dijakov, podatki o tem, kdo je koordinator, kdaj bo izvedba izvedena ter kdo jo bo izvajal, podatki o šolskem vprašalniku in ostalem materialu potrebnem za izvedbo.

V drugi datoteki so shranjeni podatki o koordinatorjih. Shranjeni so podatki o šoli, iz katere posamezen koordinator prihaja (še enkrat, ker bi bilo v nasprotnem primeru to potrebno poiskati v prvi datoteki), kontaktni podatki, podatki o udeležbi na seminarju, pridobljenih dovoljenjih staršev za izvedbo testiranja, seznam dijakov za vzorčenje (koordinator mora pripraviti seznam 15-letnih dijakov in ga posredovati nacionalnemu centru) in o prejetem priročniku z navodili o izvajanju izvedbe (koordinator iz nacionalnega centra dobi priročnik z vsemi potrebnimi navodili o tem, kako mora na šoli izvedba potekati, da je le ta v skladu z mednarodnimi pravili).

V tretji datoteki so podatki o izvajalcih, predvsem njihovi kontaktni podatki, podatki o udeležbi na seminarju (le izvajalec, ki se je udeležil seminarja, sme izvajati testiranja), priročniku (izvajalec od nacionalnega centra prejme priročnik z navodili za izvajanje raziskave), podpisani pogodbi (zaradi varovanja tajnosti mora vsak, ki pride v stik z materiali raziskave, podpisati pogodbo o varovanju tajnosti) in podatki o oddani napotnici (izvedbe opravljajo študentje pedagoških smeri, za izplačilo honorarja potrebujejo napotnico študentskega servisa).

V četrti datoteki so podatki o ravnateljih šol, ki sodelujejo pri projektu. V njej se zopet ponovijo podatki o šoli, iz katere prihaja ravnatelj, zabeleženi so ravnateljevi kontaktni podatki in podatki o udeležbi na seminarju (nacionalni center organizira seminar za ravnatelje in koordinatorje, na katerem jim podrobneje predstavi izvajanje raziskave).

V zgornjem opisu imamo torej opisane želje uporabnikov in podatke, ki so nam na voljo. Ker sama tudi poznam potrebe uporabnikov, sem predlagala, da se vodi tudi podatke o stroških izvajanja, kar bi na koncu omogočilo hiter obračun stroškov in pohitrilo izplačilo honorarjev. Ker si uporabniki ne predstavljajo dobro, kaj vse jim lahko ponudimo z novo organizacijo podatkov, bomo želje spoznavali tudi v nadaljevanju. Takrat bodo uporabniki bolj spoznali novo organizacijo podatkov in si bodo bolj predstavljali, kaj je s pomočjo programa Access mogoče izdelati. Zaradi poznavanja projekta in potreb vodenja podatkov bom skozi izdelavo baze za morebitne nove ideje znala iskati čimbolj funkcionalne rešitve tudi sama.

Ko vemo, kaj nam je na voljo, kakšne podatke želimo hraniti v podatkovni bazi in kaj zahteva uporabnik, se lotimo same izdelave baze. To je opisano v nadaljnjih poglavjih.

### **5.1.2 Razporeditev podatkov med tabele**

V tem razdelku opišimo naše razmišljanje o razdelitvi podatkov med tabele novo nastajajoče baze. Iz sedanjih podatkov moramo razbrati, kateri objekti (entitete) v našem primeru obstajajo. Imena objektov bomo navajali v poševni pisavi z enakimi imeni, kot jih bodo imele tabele, le da bomo imena tabel pisali z velikimi tiskanimi črkami. V imenih bomo izpustili šumnike in presledke. Izpustili jih bomo zgolj iz previdnosti, ker včasih povzročajo težave, čeprav nam sam program Access te omejitve ne postavlja.

Glede na to da imamo sedaj štiri datoteke s različnimi podatki, bi na prvi pogled rekli, da imamo štiri objekte: *izvedba*, *koordinator*, *ravnatelj* ter *izvajalec*. Prvi predstavlja določeno izvedbo, torej testiranje na neki konkretni šoli. Drugi objekt je koordinator izvajanja naše raziskave na določeni šoli, ravnatelj pa je ravnatelj te šole. Izvajalec kot četrti objekt je s strani nacionalnega centra določena oseba, ki poskrbi za konkretno izvedbo testiranja na določeni šoli. Za objekte *koordinator*, *ravnatelj* ter *izvajalec* beležimo njihove naslove. Opazimo, da je smiselno hraniti poštno številke in kraje kot svoj objekt. S tem jih hranimo le enkrat. Poleg tega tudi ni smiselno, da od uporabnikov zahtevamo tako vnos pošte številke kot tudi kraja, saj je s poštno številko kraj enolično določen in obratno. Ker ta odnos velja vedno, je smiselno, da v naslovih hranimo le en podatek (poštno številko). Kadar potrebujemo tudi ime kraja, preko tabele, v kateri so shranjene poštno številke in kraji, to ime izvemo. Torej je naš peti objekt *posta*.

Na vsaki šoli se izvaja izvedba za vsak učni program posebej. Pri sedanji evidenci podatkov ima vsaka izvedba svojo mednarodno določeno identifikacijsko številko in to želimo ohraniti tudi v prihodnosti. Vendar se trenutno za vsako izvedbo pod posamezno številko vodi podatke o šoli, kot so naslov, ravnatelj, koordinator in podobno. Torej se podatki šole za vsak program ponovijo. Težavo rešimo tako, da tovrstne podatke izločimo v samostojno tabelo oz. ustvarimo nov objekt. Naredimo si tabelo SOLA, kamor si bomo zabeležili splošne podatke o šoli. Naš šesti objekt je torej *sola*. Za beleženje izobraževalnih programov si naredimo tabelo PROGRAM, v katero zabeležimo vse izobraževalne programe, katerih oznako bomo navedli v tabeli IZVEDBA. V tej bomo potem beležili le podatke o posameznih izvedbah. Pri izvedbah si bomo zabeležili tudi oznako šole. Preko relacije s tabelo SOLA si bomo lahko pridobili podatke o šoli. Torej naš sedmi objekt predstavlja tabela PROGRAM. Praksa kaže, da je koordinator na šoli pogosto kar ravnatelj. To za nas pomeni, da se podatki lahko podvojijo, saj vodimo tako podatke o koordinatorju, kot tudi ravnatelju določene šole. Ker se moramo podvajanju podatkov v bazah izogibati, moramo zadevo rešiti drugače. Zato uvedemo dve tabeli OSEBA in TIPOSEBE. V tabeli OSEBA bomo vodili evidenco oseb (koordinatorjev in ravnateljjev) ter njihovih podatkov. Vsaki osebi bomo dodelili oznako iz tabele TIPOSEBE, ki nam bo povedala, koga ta oseba predstavlja (koordinatorja, ravnatelja ali osebo, ki je v obeh vlogah). Število objektov se ne spremeni, saj bo objekt *oseba* nadomeščal predhodna objekta *koordinator* in *ravnatelj*. S tem se nam število objektov celo zmanjša, vendar novo ustvarjeni objekt *tiposebe* število ravno izenači.

Izpišimo si torej vseh sedem objektov, ki smo si jih zgoraj zamislili. To so *izvedba*, *izvajalec*, *posta*, *sola*, *program*, *oseba* in *tiposebe*. Naša podatkovna baza bo torej sestavljena iz sedmih tabel. Ko vemo, katere tabele potrebujemo, jim moramo določiti polja (attribute).

Omenimo, da v teoriji podatkovnih baz obstaja postopek, imenovan normalizacija, ki razporeditev podatkov natančno določa. Postopku zaradi njegove kompleksnosti in naše razmeroma majhne količine podatkov ne bomo posvečali pozornosti.

### 5.1.3— Določanje polj v tabelah

Pri določanju polj se vprašamo, katere podatke želimo beležiti za posamezen objekt. Največja pomoč pri tem so že obstoječi podatki, ki nam povedo, kaj si je uporabnik v procesu izvajanja projekta že beležil.

Pojdimo od tabele do tabele in si naredimo načrt, katera polja naj bi posamezna tabela vsebovala:

**IZVEDBA** (StrID, SchID, SolaID, ProgramID, OsebaID, Datum1, Datum2, Ponovitev, DatumPon, Izvajalec1, Izvajalec2, ScQPoslan, ScQPrejet, StDijakov, Soglasja,

PorociloPoslano, PorociloPrejeto, VzorecPoslan, PaketOddan, PaketVrnjen, Stroski, Opombe);

**IZVAJALEC** (IzvajalecID, Ime, Priimek, Naslov, PostnaSt, Tel1, Tel2, Mail1, Mail2, Seminar, PrirocnikPrejet, Napotnica, Opombe);

**POSTA** (PostnaSt, Kraj);

**SOLA** (SolaID, ImePolno, ImeKratko, Naslov, PostnaSt, Seznam15, Opombe).

**PROGRAM** (ProgramID, Oznaka, Opis);

**OSEBA** (OsebaID, Ime, Priimek, Tel1, Tel2, Fax, Mail1, Mail2, TipID, Seminar, PrirocnikPrejet, SolaID, Opombe);

**TIPOSEBE** (TipID, TipOpis);

Kakšen podatek bo posamezno polje vsebovalo, je večinoma razvidno iz samih imen polj. Pri tistih, kjer pa pomen ni dovolj razviden, bomo v nadaljevanju, ko bomo dokončno oblikovali tabele, napisali še ustrezno pojasnilo o vsebini posameznega polja.

#### 5.1.4 Določanje ključev

Pomemben korak pri načrtovanju baze podatkov je določanje ključev. Kot smo omenili že v poglavju 2.2.2, s pomočjo ključev povezujemo podatke iz različnih tabel in poskrbimo, da so zapisi enolični. Poznamo primarni in tuji ključ. Povezavi med njima rečemo relacija. S pomočjo relacij povemo, kako lahko informacije v tabelah združimo. Več o relacijah si bomo ogledali v razdelku 5.1.5. Tu si oglejmo le enostaven primer, kaj nam relacije omogočajo. Denimo, da želimo po običajni pošti poslati dopis ravnatelju določene šole, ki sodeluje v projektu. Ko poiščemo ustreznega zapis v tabeli oseb (OSEBA), v tej tabeli ni naveden naslov šole. Naslovi so namreč zbrani v tabeli SOLA. Preko tujega ključa SolaID v tabeli OSEBA pridemo do primarnega ključa SolaID v tabeli SOLA in s tem do pripadajočega zapisa z iskanim naslovom.

Če povzamemo: primarni ključ nam omogoča, da ločimo med posameznimi zapisi v tabeli, tuji ključ pa, da se povežemo z ustreznim primarnim ključem druge tabele in s tem zapisu iz primarne tabele dodamo ustrezne podatke, ki jih hranimo v drugi tabeli.

Imenom polj, ki predstavljajo ključe (oziroma natančneje, ki so del ključa), bomo oz. smo že dali končnico ID, razen pri poštni številki. Kandidate za ključe smo tako več ali manj določili v prejšnjem koraku, ko smo določali polja. Kasneje bomo natančneje napisali, katera polja pomenijo kakšen ključ. Pri določanju ključev je pomembno, da za primarni ključ tabele vzamemo polje, katerega vrednost se ne bo nikoli ponovila. Zato je včasih za primarni ključ potrebno vzeti več polj, katerih kombinacija nam zapise enolično označi. Najbolj enostaven primer primarnega ključa je kar zaporedna številka zapisa. Vendar nam to ponavadi o posameznem zapisu ne pove kaj dosti. Tako si velikokrat želimo za primarni ključ vzeti nek podatek, ki je enoličen in nam da še dodatno informacijo o zapisu. Primer takega ključa je npr. poštna številka. Vemo, da ne obstajata dve enaki poštni številki, zato je ta podatek primeren za primarni ključ. Hkrati pa nam že sama številka natančno določi kraj.

Za vsako tabelo je torej pomembno, kaj je njen primarni ključ. Določanje tujih ključev pa je odvisno od tega, katere povezave želimo imeti med tabelami. Pojdimo torej od tabele do tabele in si podrobneje pogledjmo, kateri podatki so najprimernejši za primarne in tuje ključe v naših tabelah:

- V tabeli IZVEDBA bo primarni ključ sestavljen iz polja StrID in polja SchID, ki predstavljata mednarodno določeno, enolično oznako izvedbe. V tej tabeli



potrebujemo tudi tuje ključe in sicer SolaID, preko katerega izvemo podatke o šoli, na kateri se izvedba izvaja, ProgramID, da vemo v katerem učnem programu se izvaja in OsebaID, da iz tabele OSEBA lahko izvemo podatke o koordinatorju oziroma ravnatelju te šole. Še enkrat poudarimo, da ni nujno, da se polja, ki predstavljajo tuje ključe, imenujejo enako kot primarni ključi v drugi tabeli. Tako bi npr. v tej tabeli polje OsebaID lahko mirno poimenovali OznakaOsebe in ga kot tuji ključ povezali s primarnim ključem OsebaID v tabeli OSEBA.

- Tabela IZVAJALEC ima primarni ključ IzvajalecID, ki bo kar zaporedna številka zapisa in tuji ključ PostnaSt, ki je primarni ključ v tabeli POSTA. Z njegovo pomočjo bomo iz te tabele pridobili podatke o kraju.
- V tabeli SOLA določimo primarni ključ SolaID, ki bo s strani nacionalnega centra določeno trimesčno število ter tuji ključ PostnaSt.
- V tabeli PROGRAM bo primarni ključ polje ProgramID. To bo zaporedna številka učnega programa. Tujih ključev v tej tabeli nimamo, saj ne potrebujemo nobenih dodatnih podatkov iz drugih tabel.
- V tabeli OSEBA bo primarni ključ zaporedna številka zapisa v polju OsebaID. Polji TipID ter SolaID bosta tuja ključa, potrebna, da vemo, na kateri šoli je oseba in koga predstavlja.
- Polje TipID je primarni ključ v tabeli TIPOSEBE. Tujih ključev tukaj ne potrebujemo.

### 5.1.5 Relacije med tabelami

V prvi fazi načrtovanja smo se trudili, da se podatki v tabelah ne bi ponavljali. Po potrebi smo določene podatke združili v novo tabelo in na koncu dobili sedem tabel. Ker pa z izdelavo podatkovne baze želimo doseči, da bi imeli dostop do vseh zbranih podatkov na enem mestu, moramo tabele med seboj povezati. Možnost, da lahko uporabljamo podatke iz več kot ene tabele, je osnovna značilnost relacijske podatkovne baze. Tabele so lahko med seboj v različnih relacijah. Pri ponazarjanju teh relacij si bomo pomagali z izmišljenim primerom iz poglavja 2.2.1, ker bo zaradi enostavnosti bralcu verjetno lažje razumljiv kot pa problem, ki ga rešujemo skozi to diplomsko nalogo.

Poznamo tri vrste relacij:

- »Ena proti ena« kjer ima vsak zapis iz tabele A največ en ujemajoči se zapis v tabeli B, vsak zapis iz tabele B pa ima lahko kvečjemu en ujemajoči se zapis v tabeli A. To vrsta relacije uporabimo na primer takrat, ko želimo eno veliko tabelo razdeliti na več manjših delov. Primer take relacije imamo v našem izmišljenem primeru med tabelama ZAPOSLENI (slika 3) in NASLOVI (slika 4). Značilnost te relacije je, da sta primarna ključa obeh tabel enaka, kar pomeni, da imata enak podatkovni tip in enake podatke. V našem primeru enemu zaposlenemu pripada natanko en naslov in obratno, vsak naslov pripada natanko enemu zaposlenemu.

IDZaposlenega	ImePriimek	IDOddelka	Stopnja
MK1	Miha Kranjc	PO	IV
MH1	Maja Hrust	KO	VII
LS1	Lidija Svet	RO	V
MK2	Mitja Kern	NO	IV
ML1	Mateja Likar	PO	V

Slika 3 Slika tabele ZAPOSLENI

IDZaposlenega	Naslov	Pošta	Kraj	TelDoma
MK1	Črnuška 1	1000	Ljubljana	01/123-45-67
MH1	Prešernova 7	2000	Maribor	02/765-41-23
LS1	Hubadova 3	1000	Ljubljana	01/567-24-31
MK2	Gregorčičeva 15	3000	Celje	03/432-15-67

Slika 4 Slika tabele NASLOVI

Obe tabeli lahko združimo v eno veliko tabelo. V njej zapisom, ki v eni tabeli nimajo ustreznega zapisa, del podatkov manjka.

IDZaposlenega	ImePriimek	IDOddelka	Stopnja	Naslov	Pošta	Kraj	TelDoma
MK1	Miha Kranjc	PO	IV	Črnuška 1	1000	Ljubljana	01/123-45-67
MH1	Maja Hrust	KO	VII	Prešernova 7	2000	Maribor	02/765-41-23
LS1	Lidija Svet	RO	V	Hubadova 3	1000	Ljubljana	01/567-24-31
MK2	Mitja Kern	NO	IV	Gregorčičeva 15	3000	Celje	03/432-15-67
ML1	Mateja Likar	PO	V				

Slika 5 Slika združene tabele

- Relacija »ena proti mnogo« je najbolj običajna vrsta relacije. V tej relaciji ima lahko zapis iz tabele A veliko ujemajočih se zapisov v tabeli B, zapis iz tabele B pa ima kvečjemu en ujemajoči se zapis v tabeli A. Primer take relacije je relacija med tabelama ODDELEK (slika 6) in ZAPOSLENI (slika 7). Na enem oddelku podjetja je lahko zaposlenih več delavcev, posamezen delavec pa je lahko zaposlen le na enem oddelku. Določen oddelek je lahko v strukturi podjetja predviden, a v njem še nimamo zaposlenih. Pri tej povezavi potrebujemo že večkrat omenjeni primarni ključ (rumeni stolpec na sliki 6) in tuji ključ (rumeni stolpec na sliki 7). Če torej želimo izvedeti, v kateri sobi je Maja, preko povezave tujega ključa v tabeli ZAPOSLENI z vrednostjo KO in primarnega ključa v tabeli ODDELEK, pridemo do ustreznega zapisa v tabeli ODDELEK. Tam ugotovimo, da je oddelek s primarnim ključem KO v sobi S10N1. Maja je torej v sobi S10N1.

IDOddelka	Naziv	Lokacija
PO	Prodajni oddelek	S1N2
KO	Kadrovski oddelek	S10N1
RO	Računovodski oddelek	S3N3
NO	Nabavni oddelek	S14N2
MA	Marketing	S1N1

Slika 6 Slika tabele ODDELEK

IDZaposlenega	ImePriimek	IDOddelka	Stopnja
MK1	Miha Kranjc	PO	IV
MH1	Maja Hrust	KO	VII
LS1	Lidija Svet	RO	V
MK2	Mitja Kern	NO	IV

Slika 7 Slika tabele ZAPOSLENI

- Tretja vrsta relacije je relacija »mного proti mnogo«, kjer ima lahko zapis iz tabele A veliko ujemajočih se zapisov v tabeli B, in tudi zapis iz tabele B ima lahko veliko ujemajočih se zapisov v tabeli A. To vrsto relacije je v obstoječih sistemih za upravljanje z bazami podatkov mogoče izvesti samo tako, da vpeljemo tretjo tabelo (z imenom stična tabela). Njen primarni ključ je sestavljen iz kombinacije primarnih ključev tabel A in B. Relacijo »mного proti mnogo« torej izvedemo kot dve relaciji »ena proti mnogo«. V našem izmišljenem primeru take povezave zaenkrat nimamo. Za boljšo predstavbo naš primer razširimo. Dodajmo tabelo OPRAVILA, kjer bo

podjetje vodilo opravila, ki jih je potrebno opraviti. Še enkrat omenimo, da bo to le primer, s katerim si želimo prikazati relacijo mnogo proti mnogo in mogoče praktično ne bi bil uporaben. Recimo, da bi podjetje delalo razporeditev opravil med zaposlene. Predpostavimo, da en zaposlen lahko opravlja več opravil in tudi eno opravilo je lahko dodeljeno več zaposlenim. Med tabelo OPRAVILA in ZAPOSLENI s tem nastane relacija mnogo proti mnogo. Ustvariti moramo stično tabelo. Naj bo to tabela POKRITOST (tabela 6). Stična tabela mora vsebovati primarna ključa tabel, ki ju želimo povezati. Zato ima tabela POKRITOST primarni ključ sestavljen iz polj *IDZaposlenega* in *IDOpravila*. Tabeli ZAPOSLENI in OPRAVILA nato povežemo z relacijo ena proti mnogo s tabelo POKRITOST. V stično tabelo lahko dodamo tudi druga polja. Tako smo si dodali polje *Ure*, v katerem si bo podjetje lahko beležilo, koliko ur potrebuje posamezen delavec za posamezno opravilo. Zapisi v tabeli 6 prikazujejo, da opravilo 3 opravljata osebi MH1 (30 ur) in LS1 (4 ure), ki opravlja tudi delo 2 (20 ur).

IDZaposlenega	ImePriimek	IDOddelka	Stopnja
MK1	Miha Kranjc	PO	IV
MH1	Maja Hrust	KO	VII
LS1	Lidija Svet	RO	V
MK2	Mitja Kern	NO	IV

**Tabela 4 Tabela ZAPOSLENI**

IDOpravila	Opis
1	Naročilo izdelka x
2	Obračun plače maj
3	Zaposlitev 5 novih delavcev
4	Povečanje prodaje

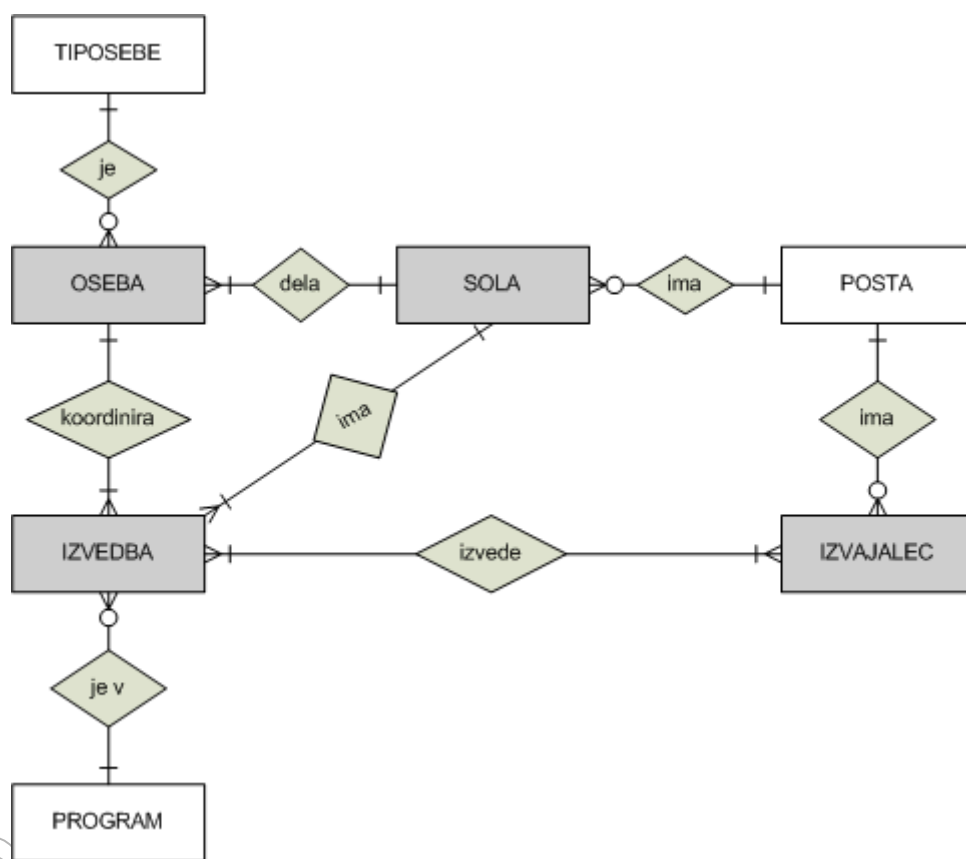
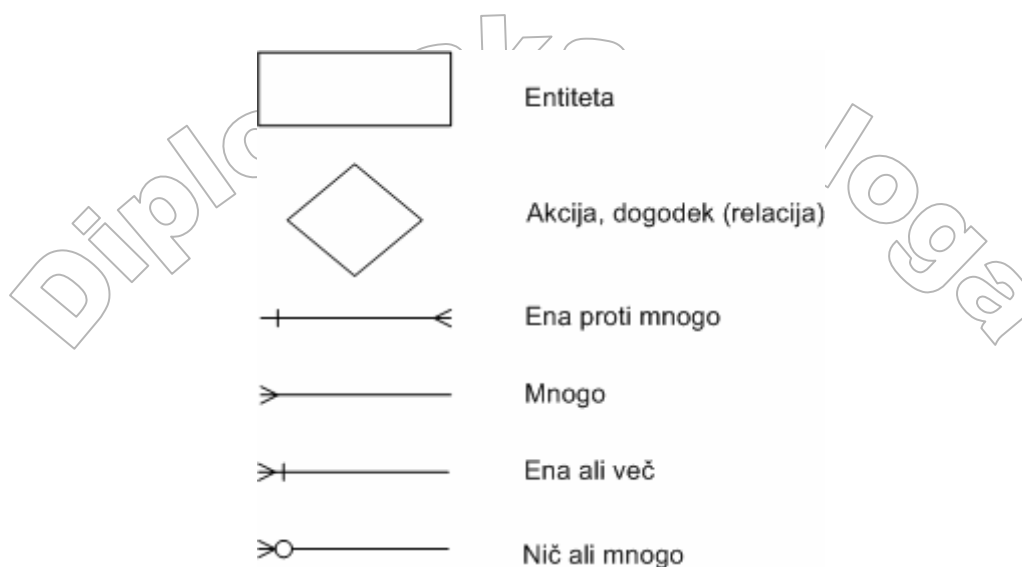
**Tabela 5 Tabela OPRAVILA**

IDOpravila	IDZaposlenega	Ure
1	MK2	1
2	LS1	20
3	MH1	30
4	MK1	16
3	LS1	4

**Tabela 6 Stična tabela POKRITOST**

Relacije lahko opredelimo tudi podrobneje. Tako lahko na primer pri relaciji ena proti mnogo vemo, da bo vsakemu zapisu ustrezal vsaj en zapis v drugi tabeli. Torej se ne more zgoditi, da za poljubni tuji ključ, ki nastopa v tej relaciji, v drugi tabeli ni zapisa, ki bi imel primarni ključ z isto vrednostjo. Tako relacijo lahko označimo z ena proti ena ali več. Če pa ni nujno, da v drugi tabeli zapis obstaja, je to potem relacija ena proti nič ali mnogo.

Sedaj, ko poznamo različne vrste relacij, si oglejmo tako imenovani model ER (slika 8) za problem, ki ga rešujemo. Z njim grafično prikažemo entitete, njihove lastnosti (attribute) in relacije med njimi. Model bomo poenostavili in prikazali le entitete in relacije med njimi, saj smo attribute navedli že v poglavju 5.1.3. Za lažje razumevanje pojasnimo simbole uporabljene v modelu ER.



Slika 8 Poenostavljen model ER

Povezave v modelu ER opišemo še z besedami. Za vsak učni program se lahko izvaja več izvedb. V projektu PISA se v tretjem ciklu za posamezni učni program izvede več testiranj dijakov. Ker se v kakšnem od prihodnjih ciklov lahko zgodi, da kateri od učnih programov ne bo imel izvedb, smo uporabili oznako »nič ali mnogo«. To pomeni, da imamo med tabelama PROGRAM in IZVEDBA relacijo »ena proti mnogo«. Enako je s povezavo med tabelo TIPOSEBE in OSEBA. Posamezen tip osebe lahko označuje več oseb, saj imamo v tabeli OSEBA lahko več ravnateljev, več koordinatorjev in verjetno tudi več ravnateljev, ki so hkrati tudi koordinatorji. Lahko pa bi se zgodilo, da ne bi imeli nobenega ravnatelja, ki je hkrati tudi koordinator. Na drugi strani pa vsaka oseba lahko nastopa le v eni od teh treh omenjenih vlog. To pomeni, da tudi med tabelo TIPOSEBE in OSEBA obstaja relacija »ena

proti mnogo«. Na enak način tabelo POSTA povežemo s tabelama IZVAJALEC in SOLA z relacijo »ena proti mnogo«. Na isti poštni številki je lahko več šol, vsaka šola pa je na natanko enem naslovu (in ima s tem eno poštno številko). Ker se projekt PISA večinoma izvaja na srednjih šolah, zagotovo lahko obstaja poštna številka, ki ne pripada nobeni šoli (srednje šole so v večjih krajih). V tabelo POSTA bomo namreč shranili vse poštno številke v Sloveniji.

Za projekt PISA želimo voditi le podatke o šolah, ki dejansko sodelujejo v nekem ciklu raziskave in ne o vseh šolah, ki obstajajo. Torej šole, na kateri ni nobene izvedbe, ne more biti v bazi. To pomeni, da bomo imeli v bazi le šole, ki imajo vsaj eno izvedbo in vsaj eno kontaktno osebo, ki bo koordinirala najmanj eno izvedbo. Predpostavimo tudi, da posamezno izvedbo koordinira le ena oseba. To zahteva povezavo med tabelama OSEBA in IZVEDBA tipa »ena proti mnogo«, kajti ena oseba lahko koordinira več izvedb. Predpostavljamo tudi, da posamezna oseba predstavlja le eno šolo. Torej ravnatelj ne more biti hkrati ravnatelj na več šolah ali morda ravnatelj na eni šoli, koordinator pa na drugi. Enako tudi za koordinatorja predpostavljamo, da je koordinator le na eni in ne more biti na več šolah. Torej med tabelama SOLA in OSEBA obstaja relacija »ena proti mnogo«, ker lahko obstaja več oseb iz ene šole ampak vsaka oseba predstavlja le eno šolo. Na posamezni šoli se zaradi različnih učnih programov lahko izvede več izvedb, zato obstaja med tabelama SOLA in IZVEDBA povezava »ena proti mnogo«. Jasno je, da se posamezna izvedba izvaja le na eni šoli.

Ostane nam le še povezava med izvedbo in izvajalcem. Iz prakse izhaja dejstvo, da en izvajalec lahko izvaja več izvedb. Zgodi se tudi, da eno izvedbo izvaja več izvajalcev, bodisi sočasno bodisi zaporedno zaradi velikega števila dijakov ali zaradi ponovitve izvedbe. To nas pripelje do relacije »mnogo proti mnogo« med tabelama IZVAJALEC in IZVEDBA. Kot smo že omenili, realizacija te relacije zahteva dodatno stično tabelo. Ustvarimo osmo tabelo IZVEDBAIZVAJALEC. Vanjo moramo dati primarna ključa iz tabel IZVEDBA in IZVAJALEC. Ta tabela bo v praksi predstavljala konkretno izvedbo z določenim izvajalcem. Torej je smiselno vanjo prestaviti še nekaj podatkov iz tabele IZVEDBA in sicer polja Datum1, Datum2, Ponovitev, DatumPon, Izvajalec1, Izvajalec2. Ampak ali se ne bi dalo zadeve poenostaviti? Izvedba se lahko ponavlja in izvajajo jo lahko različni izvajalci. Uporabnik si za vsako tako izvedbo želi beležiti, kdo jo je izvedel in kdaj. To bomo dosegli, če primarnemu ključu dodamo še četrto polje IzvedbaID, ki bo označevalo, katera izvedba po vrsti je to. S tem se zmanjša število ostalih polj. Ker bo sedaj vsaka izvedba imela enolično oznako, čeprav bo isti izvajalec izvedbo večkrat ponovil, si lahko beležimo čas izvedbe v enem samem polju Datum. Potrebujemo še podatek, ob kateri uri bo izvedba. Zato dodamo polje Ura. Polja Ponovitev, DatumPon, Izvajalec1 in Izvajalec2 postanejo s tem nepotrebna in jih odstranimo. Hkrati smo s tem rešili problem, če bi slučajno prišlo do več kot ene ponovitve izvedbe, saj prej te nove ponovitve ne bi mogli zapisati v tabelo. Tudi polja PaketOddan, PaketVrnjen, Stroski bo bolj smiselno imeti v stični tabeli. Za morebitne opombe si tudi v stični tabeli dodajmo polje Opombe. Poglejmo si torej kakšne attribute bodo imele tabele, po uvedbi stične tabele.

**IZVEDBAIZVAJALEC** (StrID, SchID, IzvajalecID, IzvedbaID, Datum, Ura, PaketOddan, PaketVrnjen, Stroski, Opombe)

**IZVEDBA** (StrID, SchID, SolaID, ProgramID, OsebaID, ScQPoslan, ScQPrejet, StDijakov, Soglasja, PorociloPoslano, PorociloPrejeto, VzorecPoslan, PaketOddan, PaketVrnjen, Stroski, Opombe);

**IZVAJALEC** (IzvajalecID, Ime, Priimek, Naslov, PostnaSt, Tel1, Tel2, Mail1, Mail2, Seminar, PrirocnikPrejet, Napotnica, Opombe);

Tabeli IZVEDBA in IZVAJALEC bosta tako v relaciji »ena proti mnogo« s tabelo IZVEDBAIZVAJALEC.

S tem smo povezali vse naše doslej načrtovane tabele. V naslednjem poglavju bomo opisali, kakšne spremembe so bile potrebne potem, ko smo načrt pregledali skupaj z uporabniki. Dodali bomo morebitne popravke in narisali dokončni model ER, kamor bomo vključili tudi stično tabelo IZVEDBAIZVAJALEC. Vse te povezave bomo kasneje seveda realizirali s pomočjo programa Access.

### 5.1.6 Pregled načrta skupaj z uporabnikom

Ko imamo izdelan načrt podatkovne baze, ga je potrebno skupaj z uporabnikom natančno pregledati. S tem se odkrijejo morebitne pomanjkljivosti in neskladja med uporabnikovimi željami in našimi predvidenimi rešitvami. V našem primeru smo torej izdelani načrt pregledali skupaj z uporabniki (sodelavci pri projektu PISA) in poskušali ugotoviti, če so v njem zajete vse potrebne informacije. Pri tem smo ugotovili, da bi uporabniki želeli beležiti tudi podatek, v katero območno enoto Zavoda za šolstvo spada določena šola. Zato je potrebno dodati novo polje v tabelo SOLA. Zaradi večje fleksibilnosti bomo raje dodali novo tabelo OBMOČJE in v njej dve polji: ObmocjeID in Enota. Z relacijo »ena proti mnogo« jo bomo povezali s tabelo SOLA. Eno območje namreč lahko vsebuje več šol, posamezna šola pa seveda spada le v eno območje. S tem smo dobili deveto tabelo naše bodoče baze.

Uporabniki so bili navdušeni nad dodanim poljem za beleženje stroškov in dobili nove ideje. Zato smo morali dodati še dodatna polja za bolj specifično beleženje stroškov. Polje Stroški smo nadomestili z novimi polji StrošekIzvedba, PrevozenihKM, Cestnina in Drugo. V polju StrošekIzvedba bo shranjen znesek honorarja za izvajalca. Polje PrevozenihKM nam bo omogočalo hranjenje podatka, koliko kilometrov je prevozil izvajalec in polje Cestnina morebitno plačano cestnino. V polje Drugo bo možno zabeležiti stroške parkiranja, žetonov ipd. Končna vsebina tabele je:

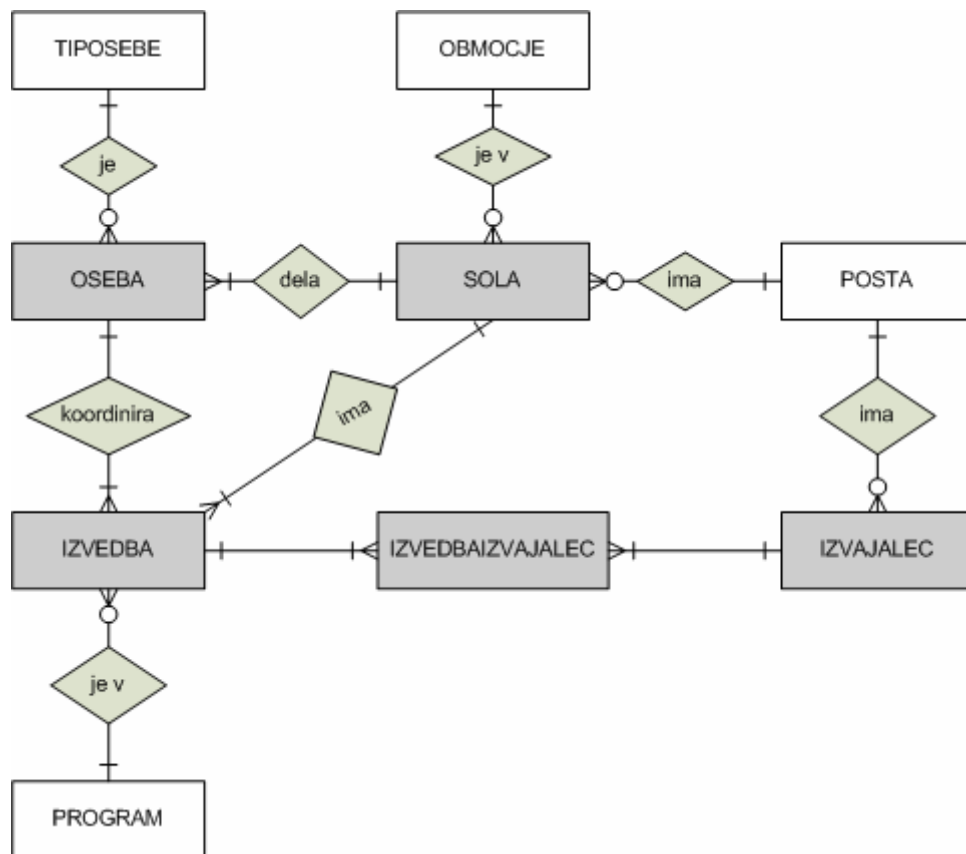
**IZVEDBAIZVAJALEC** (StrID, SchID, IzvajalecID, IzvedbaID, Datum, Ura, PaketOddan, PaketVrnjen, StrošekIzvedba, PrevozenihKM, Cestnina, Drugo, Opombe).

Prav tako je potrebno beležiti, ali je bil koordinatorju poslan priročnik. Zato v tabelo OSEBA dodamo polje PriročnikPoslan, kjer bo mogoče zabeležiti datum, kdaj je bil priročnik poslan. Če datum ne bo naveden, se ve, da priročnik še ni bil poslan. Koordinator mora prejeti priročnik z natančnimi navodili o poteku raziskave na šoli. Prvotno polje PriročnikPrejet je namenjen beleženju, ali je koordinator priročnik prejel. Od koordinatorja se namreč zahteva, da prejetje priročnika potrdi po elektronski pošti. Če ga ne prejme, ga nacionalni center pošlje še enkrat.

Uporabniki iz izkušenj vedo, da je smiselno pri izvajalcih beležiti tako stalni kot začasni naslov. Izvajalci so namreč študentje, ki stanujejo izven domačega kraja in želijo pošto prejemati na začasni naslov. Zato bomo v tabeli IZVAJALEC dodali polji NaslovZacasni in PostnaStZac. Uporabnika lahko zanima tudi, ali ima izvajalec avto, če je podpisal pogodbo in če je vrnil rezervni paket. To pomeni še tri nova polja v tabeli IZVAJALEC in sicer Avto, Pogodba in RPaketVrnjen. Informacija o avtu je koristna za planiranje in razdeljevanje izvedb izvajalcem. Tako se lažje ve, koga se lahko pošlje v bolj oddaljene kraje in kdo mora izvedbe opravljati v bližini bivališča. Polje PriročnikPrejet lahko odstranimo. Uporabniki so namreč ugotovili, da tega podatka ni smiselno voditi. Izvajalci namreč prejmejo priročnike na seminarju. Tako se že iz prisotnosti na seminarju ve, kdo je dobil priročnik in kdo ne. Nacionalni center organizira seminar za izvajalce v Ljubljani, Mariboru in Kopru, zato si prisotnost beležimo v polju Seminar z zapisom oznak LJ, MB ali KP. Iz tega vemo, kdo je bil prisoten in kje.

Na uporabnikovo željo odstranimo tudi polje PorociloPrejeto v tabeli IZVEDBA. Prvotno smo namreč nameravali beležiti, če je naslovnik prejel poročilo o posamezni izvedbi, a se je pokazalo, da to ni potrebno.

Skupaj si ogledjmo dokončen model ER (slika 9) z vsemi potrebnimi tabelami in povezavami. Dodali smo torej tabeli OBMOCJE in IZVEDBAIZVAJALEC. Tabelo OBMOCJE povežemo z relacijo »ena proti mnogo« s tabelo SOLA, saj, kot smo že omenili, eno območje vsebuje več šol. Posamezna šola seveda spada le v eno območje. Uporabili smo oznako »nič ali mnogo«, ker bi se lahko zgodilo, da iz nekega območja ne bi sodelovala nobena šola. Tabelo IZVEDBAIZVAJALEC smo povezali s tabelama IZVAJALEC in IZVEDBA z relacijo »mnogo proti ena«



Slika 9 Dokončen model ER

Po natančnem pregledu smo skupaj z uporabniki dokončno določili tudi podatkovne tipe polj v tabelah. Dokončna polja in njihove tipe prikazujejo spodnje razpredelnice. Podatkovni tip je značilnost polja, ki določa, kakšne podatke je mogoče vanj shraniti. Velikost polja (FieldSize) je lastnost, ki določa maksimalno velikost shranjenih podatkov za podatkovne tipe Besedilo, Število in Samoštevilo. Vedno poskušamo določiti najmanjšo velikost, ki še zadošča našim potrebam. S tem prihranimo pomnilnik in pohitrimo delovanje baze. Podrobnosti bomo opisali pri vsaki posamezni tabeli. V tabelah bomo označili tudi primarne ključe, tako da bo ime polja, ki predstavlja primarni ključ tabele, podčrtano.

Tabela	Polje	Podatkovni tip	Velikost polja
<b>PROGRAM</b>			
	<u>ProgramID</u>	Besedilo	1
	Oznaka	Besedilo	4

	Opis	Besedilo	255
--	------	----------	-----

**Tabela 7 Tabela PROGRAM z ustreznimi polji in njihovimi podatkovnimi tipi**

Za vsa tri polja v tabeli PROGRAM smo uporabili podatkovni tip Besedilo, ki omogoča vnos največ 255 znakov. Vendar je 255 znakov preveč, zato smo velikosti polj omejili. Za polje ProgramID je dovoljen en znak. V to polje bomo vnesli zaporedno številko izobraževalnega programa, kar zagotovo ne bo preseгло vrednosti 9. Polje Oznaka smo omejili na največ 4 znake, saj je oznaka programa sestavljena iz dveh, treh ali štirih znakov (OŠ, NPI, SPI, STSI, GIMG, GIMS). Za polje Opis smo pustili maksimalno dolžino 255 znakov, ker opisi nimajo natančno določene dolžine. Vemo pa, da bo 255 znakov zadostovalo. Na primer opis oznake NPI bi bil »Program nižjega poklicnega izobraževanja«.

Tabela	Polje	Podatkovni tip	Velikost polja
<b>TIPOSEBE</b>			
	<u>TipID</u>	Besedilo	1
	TipOpis	Besedilo	50

**Tabela 8 Tabela TIPOSEBE z ustreznimi polji in njihovimi podatkovnimi tipi**

Tudi v tabeli TIPOSEBE smo obema poljema določili podatkovni tip Besedilo. Polje TipID smo omejili na dolžino enega znaka, saj imamo trenutno le tri tipe oseb. V polju bomo beležili zaporedno številko tipa osebe. To tudi v prihodnosti ne bo preseгло števila 9, saj v bazi zagotovo ne bomo hranili več kot 10 različnih tipov oseb. Polje TipOsebe bo vsebovalo opis tipa osebe in ne bo presegal dolžine 50 znakov.

Mogoče je smiselno pojasniti, zakaj za polja, v katerih bomo hranili številske vrednosti, uporabljamo podatkovni tip Besedilo. Obstaja namreč tudi podatkovni tip Število, ki je namenjen shranjevanju številskih podatkov, s katerimi lahko kasneje računamo ali pa nad njimi izvajamo različne matematične operacije. Ker z nobenim od zgoraj naštetih polj ne bomo računali, je enostavneje, če vrednosti 1, 2, 3, ... obravnavamo kot znake.

Tabela	Polje	Podatkovni tip	Velikost polja
<b>IZVEDBA</b>			
	<u>StrID</u>	Besedilo	2
	<u>SchID</u>	Besedilo	3
	SolaID	Besedilo	3
	ProgramID	Besedilo	1
	OsebaID	Samoštevilo	Dolgo celo število
	VzorecPoslan	Datum/Čas	
	StDijakov	Število	Bajt
	ScQPoslan	Datum/Čas	
	ScQPrejet	Datum/Čas	
	Soglasja	Datum/Čas	
	PorociloPoslano	Datum/Čas	
	Opombe	Besedilo	255

**Tabela 9 Tabela IZVEDBA z ustreznimi polji in njihovimi podatkovnimi tipi**

Polja StrID in SchID sestavljata primarni ključ. StrID je mednarodno določeno dvomestno število, zato smo polje omejili na dva znaka. SchID predstavlja mednarodno določeno



tromestno število. V vsakem polju se vrednosti lahko ponovijo, zato šele oba skupaj enolično določata zapis v tabeli. Naslednja tri polja s končnico ID so tuji ključi. Ker imajo enako ime kot ustrezni primarni ključi, jih bomo podrobneje opisali pri tabelah, kjer nastopajo v vlogi primarnega ključa. Polje VzorecPoslan je tipa Datum/Čas, ki nam omogoča vnos datuma in ure. Poskrbi za pravilno razvrščanje datumov. V tem polju nameravamo hraniti datum, kdaj smo koordinatorju poslali seznam izbranih dijakov. V naslednjem polju StDijakov si bomo zabeležili število izbranih dijakov. Tokrat bomo uporabili podatkovni tip Število, saj bomo želeli s tem številom računati (npr. sešteti število izbranih dijakov v nekem programu itd.). Velikost polja »Bajt<sup>1</sup>« nam dovoljuje vnos celih števil od 0 do 255. Število izbranih dijakov za posamezno izvedbo zagotovo ne bo presegalo števila 255, zato nam ta velikost zadostuje. Polja ScQPoslan, ScQPrejet, Soglasja in PorociloPoslano so namenjena hranjenju datumov, ko smo na šolo poslali šolski vprašalnik in kdaj ga je šola prejela. Šola, ki sodeluje v raziskavi, mora namreč prejeti tudi vprašalnik za šolo. Datum v polju Soglasja bo povedal, kdaj smo na šolo poslali obrazce za dovoljenje staršev, na katerih starši označijo ali dovolijo sodelovanje njihovih otrok v raziskavi ali ne. V polje PorociloPoslano si bomo zabeležili, kdaj smo poslali poročilo. Zadnje polje Opombe bomo uporabili za beleženje morebitnih posebnosti.

Tabela	Polje	Podatkovni tip	Velikost polja	
<b>OSEBA</b>				
		<u>OsebaID</u>	Samoštevilo	Dolgo celo število
		Ime	Besedilo	25
		Priimek	Besedilo	50
		Tel1	Besedilo	11
		Tel2	Besedilo	12
		Fax	Besedilo	12
		Mail1	Besedilo	50
		Mail2	Besedilo	50
		TipID	Besedilo	1
		Seminar	Datum/Čas	
		PrirocnikPoslan	Datum/Čas	
		PrirocnikPrejet	Da/Ne	
		SolaID	Besedilo	3
		Opombe	Besedilo	255

**Tabela 10 Tabela OSEBA z ustreznimi polji in njihovimi podatkovnimi tipi.**

Polje OsebaID je primarni ključ tabele OSEBA. Podatkovni tip Samoštevilo pomeni, da se za vsak dodan zapis v polje OsebaID samodejno vnese enolično zaporedno število. Števila se ne da spremeniti ali izbrisati. To nam zagotavlja, da je primarni ključ ves čas enoličen. Tovrstni tip uporabimo vedno, ko potrebujemo enolično identifikacijo zapisov in nimamo potrebe, da bi bila enolična identifikacija sestavljena na drugačen način. Če poenostavimo: kadar nam ustreza, da so zapisi identificirani z zaporednim številom (1, 2, 3, ...), je običajno najenostavnejše imeti kot primarni ključ polje tipa Samoštevilo. Če želimo sami skrbeti za enolične oznake ali pa so te oznake za vsak zapis določene na drugačen način (npr. v polju

<sup>1</sup> Bralcu, ki mu zveni beseda bajt neslovensko in bi pričakoval besedo zlog, namenimo pojasnilo. Velikost polja označujem z besedami, ki jih uporablja slovenska različica programa Access in od tukaj tudi beseda »bajt«, ki je v [3] tudi eden izmed prevodov angleške besede »byte«.

oseba bi kot enolično identifikacijo lahko uporabili enotno matično številko – EMŠO), pa tipa Samoštevilo ne moremo uporabiti. Uporabniki v našem primeru nimajo zahtev glede oznake za določeno osebo, zato smo uporabili Samoštevilo. Pomen ostalih polj je razviden že iz imen. Povejmo le pojasnilo za polja Seminar, PriročnikPoslan, PriročnikPrejet. V polje Seminar si bomo beležili datum, kdaj se je oseba udeležila seminarja. Nacionalni center namreč organizira tudi seminarje za koordinatorje in ravnatelje, kjer se še dodatno pojasni pravila raziskave PISA. Koordinator mora prejeti tudi priročnik, zato si bomo zabeležili datum, kdaj smo ga mu poslali, v polje PriročnikPoslan in v polju PriročnikPrejet označili, če ga je prejel.

Tabela	Polje	Podatkovni tip	Velikost polja
<b>POSTA</b>			
	<u>PostnaSt</u>	Besedilo	4
	Kraj	Besedilo	100

**Tabela 11 Tabela POSTA z ustreznimi polji in njihovimi podatkovnimi tipi**

Za polje PostnaSt smo izbrali podatkovni tip Besedilo, saj s temi številkami ne bomo računali. Dobro se je zavedati še ene lastnosti tega podatkovnega tipa in sicer način razvrščanja. Izbrani tip nam bo številke razvrščal kot nize znakov (torej v vrstnem redu 10, 100, 20, 200 ) in ne kot številske vrednosti (10, 20, 100, 200). Razlika v našem primeru ne bo opazna, ker so vse poštno številke štirimestne.

Torej za števila, s katerimi ne bomo računali, v podatkovni bazi izberemo podatkovni tip Besedilo. Podobno velja tudi za telefonske številke, še posebej zato, ker pri njih ponavadi potrebujemo tudi druge znake. Omrežno skupino namreč običajno ločimo od ostalega dela z oklepajem ali poševnico ter za boljši pregled številko s presledki ali pomišljaji razdelimo na več delov. Tako bomo tudi mi shranjevali številko fiksnega telefonskega priključka v obliki 00/000-00-00 in mobilno številko v obliki 000/000-000.

Tabela	Polje	Podatkovni tip	Velikost polja
<b>IZVAJALEC</b>			
	<u>IzvajalecID</u>	Samoštevilo	Dolgo celo število
	Ime	Besedilo	25
	Priimek	Besedilo	50
	NaslovStalni	Besedilo	50
	PostnaSt	Besedilo	4
	NaslovZacasni	Besedilo	50
	PostnaStZac	Besedilo	4
	Tel1	Besedilo	11
	Tel2	Besedilo	12
	Mail1	Besedilo	50
	Mail2	Besedilo	50
	Seminar	Besedilo	2
	Napotnica	Da/Ne	
	Avto	Da/Ne	
	Pogodba	Da/Ne	
	RPaketVrnjen	Da/Ne	
	Opombe	Besedilo	255

**Tabela 12 Tabela IZVAJALEC z ustreznimi polji in njihovimi podatkovnimi tipi**

V tabeli 12 za polja Napotnica, Avto, Pogodba in RPaketVrnjen uporabimo podatkovni tip Da/Ne, ki nam omogoča beleženje katerihkoli dveh med seboj izključujočih si stanj (da/ne, prav/narobe itd.). Pri ostalih poljih ni posebnosti.

Tabela	Polje	Podatkovni tip	Velikost polja
<b>SOLA</b>			
	<u>SolaID</u>	Besedilo	3
	ImePolno	Besedilo	255
	ImeKratko	Besedilo	50
	Naslov	Besedilo	50
	PostnaSt	Besedilo	4
	Seznam15	Da/Ne	
	ObmocjeID	Besedilo	1
	Opombe	Besedilo	255

**Tabela 13 Tabela SOLA z ustreznimi polji in njihovimi podatkovnimi tipi**

Primarni ključ v tabeli SOLA je polje SolaID s podatkovnim tipom Besedilo in velikostjo polja 3. Tukaj nismo uporabili tipa Samoštevilko, ker želi nacionalni center določiti šolam trimestrna števila. Prvo mesto bo označevalo območje Zavoda za šolstvo in ostali dve zaporedno številko šole v območni enoti. Zaradi prvega znaka polje ObmocjeID načeloma ne bi bilo potrebno. A zaradi enostavnejše izvedbe povezave med to tabelo in tabelo OBMOČJE ga vseeno uporabimo. Polje Seznam15 pove, ali je šola v nacionalni center poslala seznam vseh 15-letnih dijakov na šoli. Nacionalni center namreč zbere informacije o vseh 15-letnih dijakih in nato izbere le določeno število tistih, ki kasneje v raziskavi dejansko sodelujejo.

Tabela	Polje	Podatkovni tip	Velikost polja
<b>OBMOČJE</b>			
	<u>ObmocjeID</u>	Besedilo	1
	Enota	Besedilo	50

**Tabela 14 Tabela OBMOČJE z ustreznimi polji in njihovimi podatkovnimi tipi**

Primarni ključ ObmocjeID v tabeli OBMOČJE bo enomestno število. Obstaja 9 območnih enot Zavoda RS za šolstvo, zato nam to zadostuje.

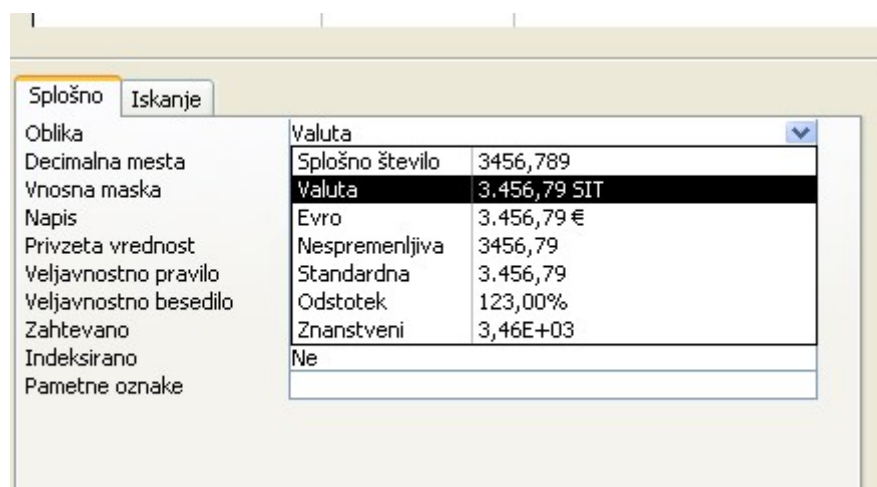
Tabela	Polje	Podatkovni tip	Velikost polja
<b>IZVEDBAIZVAJALEC</b>			
	<u>StrID</u>	Besedilo	2
	<u>SchID</u>	Besedilo	3
	<u>IzvajalecID</u>	Samoštevilko	Dolgo celo število
	<u>IzvedbaID</u>	Število	Bajt
	Datum	Datum/Čas	
	Ura	Datum/Čas	
	StrosekIzvedba	Valuta	
	PrevozenihKM	Število	Celo število

	Cestnina	Valuta	
	Drugo	Valuta	
	PaketOddan	Datum/Čas	
	PaketVrnjen	Datum/Čas	
	Opombe	Besedilo	255

**Tabela 15 Tabela IZVEDBAIZVAJALEC z ustreznimi polji in njihovimi podatkovnimi tipi**

Stična tabela IZVEDBAIZVAJALEC ima primarni ključ sestavljen iz primarnih ključev tabel, ki jih stikamo. Torej je njen primarni ključ sestavljen iz polj StrID in SchID tabele IZVEDBA ter iz polja IzvajalecID tabele IZVAJALEC. V ključ smo dodali še četrto polje IzvedbaID, ki bo označevalo zaporedno številko izvedbe. Osnovna izvedba bo imela številko 1, morebitna prva ponovitev številko 2 itd. Tip tega polja bo Številko, saj bomo na tej vrednosti včasih uporabili kako aritmetično operacijo. Zanimalo nas bo na primer število izvedb, ki imajo to polje večje od 1 ali kaj podobnega. Zakaj smo dodali to polje v primarni ključ, smo opisali v poglavju 5.1.5.

Za denarne zneske bomo uporabili podatkovni tip Valuta, ki prepreči napačno zaokroževanje med izračunavanjem. Omogoča hranjenje 15 števk pred decimalno vejico in 4 števke za decimalno vejico. Določimo mu lahko tudi obliko, v kateri želimo, da se zneski hranijo in oznako valute. Različne oblike nam samodejno postavijo različna ločila, kot so vejice in pike. Mi bomo zaenkrat izbrali obliko Valuta (slika 10), ki je prirejena slovenski valuti in dodaja končnico SIT. Kasneje pa bo uvedba Evra zahtevala izbiro oblike Evro.

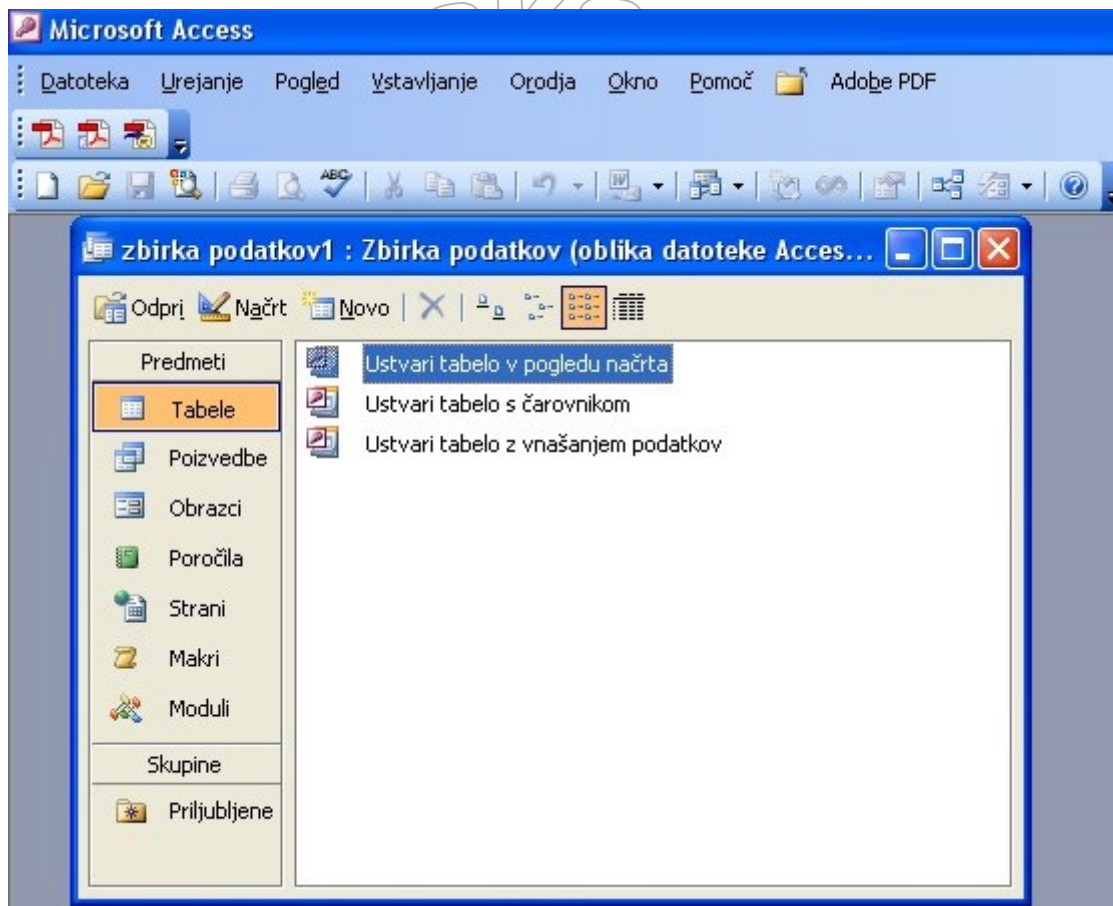


**Slika 10 Možne oblike podatkovnega tipa Valuta**

## 5.2 Izdelava tabel

V tem razdelku bomo opisali, kako prej načrtovane tabele v programu Access izdelamo. Vse tabele izdelamo znotraj prazne podatkovne baze. Slednjo ustvarimo po zagonu programa Access z izbiro »Prazna zbirka podatkov«. Pri tem moramo najprej določiti ime podatkovne zbirke, ki jo ustvarjamo.

Zaženemo torej Access, izberemo Datoteka/Nova in bazo poimenujmo »PisaMS06« kar označuje projekt »PISA Main Study 2006«. V programu Access ustvarjena podatkovna baza ob shranjevanju dobi končnico .mdb. Nato moramo ustvariti tabele, ki smo jih načrtovali. Kot vidimo na sliki 11, imamo tri možnosti. Vsako si pogledjmo nekoliko podrobneje.



Slika 11: Načini ustvarjanja tabel

### 5.2.1 Izdelava tabel v pogledu načrta

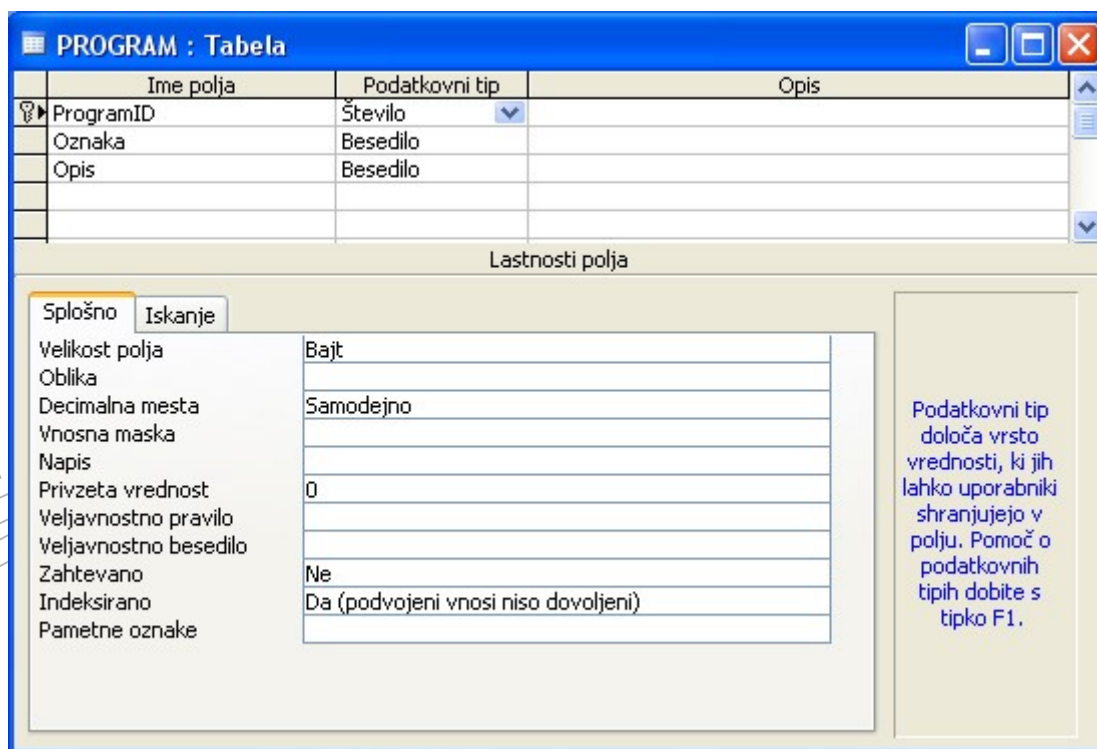
Pri tem načinu izdelave tabel se odpre okno (slika 12), kjer v zgornjem delu okna vpišemo imena polj in jim določimo podatkovni tip. V spodnjem delu določamo lastnosti posameznega polja. To so množice značilnosti, ki priskrbijo dodaten nadzor nad tem, kako so v polju shranjeni podatki, kako jih vanj vnašamo ali kako so v polju podatki prikazani. Katere lastnosti so na voljo, je odvisno od podatkovnega tipa polja. Na sliki 12 vidimo lastnosti, ki so na voljo za podatkovni tip Število. Nekatere podrobneje pojasnimo:

- »Velikost polja« smo spoznali v poglavju 5.1.6.
- »Oblika« je lastnost polja, ki določa, v kakšni obliki se podatki iz polja prikazujejo in natisnejo. Nastavitve na same shranjene podatke ne vpliva.
- »Decimalna mesta« je lastnost polja, s katero določimo, koliko decimalnih mest želimo.
- »Vnosna maska« določa v kakšni obliki želimo podatke vnašati in hraniti. Za razliko od »Oblike« se tukaj podatki tudi shranijo v obliki, ki jo definiramo.
- »Napis« je ime polja, oziroma bolj rečeno njegova oznaka. Uporabimo ga za prikaz na obrazcih in poročilih. Lahko je drugačno od imena v tabeli. Tako lahko npr. polju PostnaSt določimo napis »Poštna številka«. V vseh poročilih in izpisih bo uporabnik videl le napis »Poštna številka«, čeprav bomo pri sklicevanju na to polje znotraj programa uporabljali »pravo« ime PostnaSt. Če to polje pustimo prazno, se v obrazcih in poročilih kot oznaka uporabi "pravo" ime polja.
- »Privzeta vrednost« nam omogoča, da polju nastavimo privzeto vrednost. Lastnost je uporabna za polja, kjer je podatek za vsak zapis (ali za večino zapisov) enak, da ni potrebno vedno znova vnašati enake vrednosti. Seveda pa polju lahko privzeto

vrednost že ob vnosu zapisa ali pa kasneje spremenimo na poljubno vrednost. Pri nas to lastnost lahko uporabimo za polje IzvedbaID v tabeli IZVEDBAIZVAJALEC in jo nastavimo na 1, saj vsak izvajalec večino izvedb izvaja le enkrat. Če temu ni tako, vnesemo novo vrednost. S tem si olajšamo vnašanje podatkov. Pozorni smo torej le na tiste zapise, katerih vrednost ni enaka privzeti vrednosti in le to popravimo.

- »Veljavnostno pravilo« je lastnost polja, ki preveri, če vnešeni podatek ustreza določenim omejitvam. Nastavimo lahko torej pravilo, ki ga program Access pred shranjevanjem podatka preveri. Če vnešen podatek ne ustreza pravilu, nas opozori. Pri nas bi bila uporaba smiselna npr. za vnos datuma izvedbe. S pravilom bi veljavni datum omejili na obdobje, v katerem se izvedbe izvajajo (od 13.3. do 21.4.). S tem bi preprečili vnos napačnega datuma.
- »Veljavnostno besedilo« je besedilo, s katerim nas program Access opozori, če vnos v polje ne ustreza veljavnostnemu pravilu.
- »Zahtevano« je lastnost polja, s katero povemo, če je vnos nujen, ali pa lahko polje pustimo prazno.
- »Indeksirano« je lastnost polja, ki programu Access pomaga pri iskanju in razvrščanju zapisov. Smiselno je indeksirati polja po katerih pogosto iščemo ali po katerih podatke sortiramo. Vsak primarni ključ je avtomatično indeksiran.

Za nekatera polja že vemo, katere od naštetih lastnosti nam bi prišle prav. Te bomo navedli pri izdelavi posamezne tabele. Smiselnost nekaterih lastnosti pa bomo mogoče opazili šele kasneje pri izdelavi obrazcev v poglavju 5.4. in zato jih bomo podrobneje opisali tam.



Slika 12: Okno za izdelavo tabel v pogledu načrta

### 5.2.2 Izdelava tabel s čarovnikom

Čarovnik nam omogoča izdelavo tabel s pomočjo predhodno narejenih vzorcev tabel. Izberemo lahko polja (slika 13) iz različnih tabel, ki so podobna tistim, ki jih želimo definirati v naši tabeli. Poljem priredimo ustrezna imena. V nadaljnjih korakih nam čarovnik omogoča nastavitve primarnega ključa, ustvarjanje relacij med tabelami itd. Na koncu nam

lahko naredi tudi obrazec za vnos podatkov v pravkar ustvarjeno tabelo. Ta izbira je smiselna takrat, ko izdelujemo kakšno od »standardnih« baz podatkov, kot so različne evidence oseb, kupcev, prodajalcev in podobno. Mi tega načina ne bomo uporabljali, saj smo si lastnosti že predhodno dovolj natančno definirali in pomoči čarovnika ne potrebujemo.



Slika 13 Čarovnik za izdelavo tabel

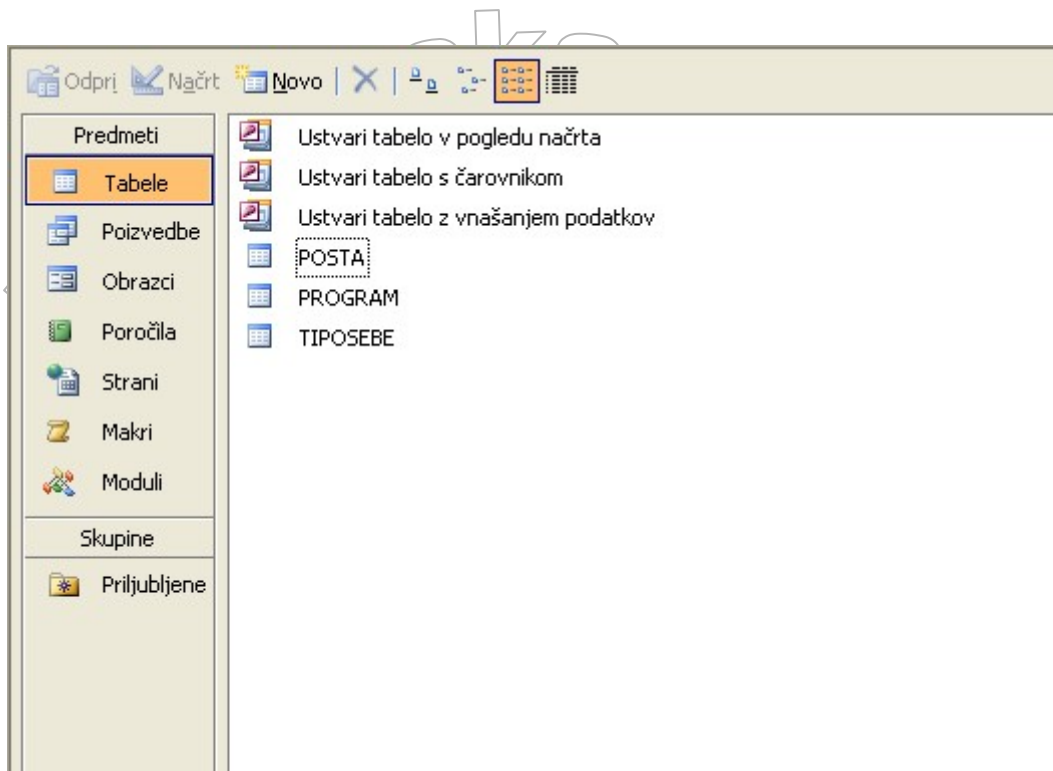
### 5.2.3 Izdelava tabel z vnašanjem podatkov

Pri tem načinu izdelave tabel vnašamo podatke v tabelo, ki nam jo program ponudi (slika 14). Pri tem skuša program sam ugotoviti ustrezne podatkovne tipe polj. Tudi tega načina ne bomo uporabljali, saj smo si tipe polj natančno definirali in ni potrebno, da jih ugotavlja program Access.

	PostnaSt	Kraj	Polje3	Polje4	Polje5
	1355	Polhov Gradec			
	4228	Železniki			
	1000	Ljubljana			

Slika 14 Izdelava tabele z vnašanjem podatkov

Pri spoznavanju možnosti smo z vsakim od načinov izdelali po eno tabelo, ki jih sedaj vidimo v osnovnem oknu (slika 15).



**Slika 15 Osnovno okno z izdelanimi tabelami POSTA, PROGRAM in TIPOSEBE.**

Ostale tabele bomo naredili v prvem načinu - pogledu načrta, ker natančno vemo, kakšne tabele potrebujemo in pomoči programa Access ne potrebujemo. Pri izdelavi si pomagamo z razpredelnicami, ki smo jih ustvarili v poglavju 5.1.6. Zaradi velikega števila atributov ne moremo pokazati lastnosti vseh polj in bomo omenili le najpomembnejše. Podrobnosti posameznih polj naj si bralec po želji ogleda na k diplomski nalogi priloženi zgoščenki.

Poglejmo si torej še izdelavo ostalih tabel. Ponekod bomo opisali tudi najpomembnejše lastnosti.

fiziko  
in  
matematiko za  
Fakulteta



Ime polja	Podatkovni tip	Opis
IzvajalecID	Samoštevilo	Zaporedna številka vnešenega izvajalca.
Ime	Besedilo	Ime izvajalca z velikimi črkami.
Priimek	Besedilo	Priimek izvajalca z velikimi črkami.
NaslovStalni	Besedilo	Stalni naslov izvajalca.
PostnaSt	Besedilo	Poštna številka stalnega prebivališča.
NaslovZacasni	Besedilo	Začasni naslov izvajalca.
PostnaStZac	Besedilo	Poštna številka začasnega prebivališča.
Tel1	Besedilo	Mobilna številka izvajalca.
Tel2	Besedilo	Stacionarna številka izvajalca.
Mail1	Besedilo	Elektronski naslov izvajalca.
Mail2	Besedilo	Elektronski naslov izvajalca.
Seminar	Besedilo	Oznaka LJ, KP ali MB ki pove kje je bil izvajalec na seminarju.
Napotnica	Da/Ne	Ali je izvajalec prinesel napotnico?
Avto	Da/Ne	Ali ima izvajalec avto?
Pogodba	Da/Ne	Ali je izvajalec podpisal pogodbo o varovanju podatkov?
RPaketVrnjen	Da/Ne	Ali je izvajalec vrnil rezervni paket?
Opombe	Besedilo	Prostor z vpis posebnosti.

Lastnosti polja	
Velikost polja	11
Oblika	
Vnosna maska	{099\ 999\}-999
Napis	Mobi
Privzeta vrednost	
Veljavnostno pravilo	
Veljavnostno besedilo	
Zahtevano	Ne
Dovoli ničelno dolžino	Da
Indeksirano	Ne
Stiskanje Unicode	Da
Način IME	Brez nadzora
Stavčni način IME	Brez
Pametne oznake	

Podatkovni tip določa vrsto vrednosti, ki jih lahko uporabniki shranjujejo v polju. Pomoč o podatkovnih tipih dobite s tipko F1.

Slika 16 Pogled načrta tabele IZVAJALEC

Tabela IZVAJALEC (slika 16) vsebuje dve polji za hranjenje telefonskih števil Tel1 in Tel2. V polje Tel1 bomo shranjevali mobilne številke, v polje Tel2 pa stacionarne številke. Na sliki 16 je vidna vnosna maska za mobilno številko. Vnosno masko definiramo s pomočjo 13 oznak. To so 0, 9, #, L, ?, A, a, &, C, >, <, ! in \. Vsaka ima točno določen pomen. Razložimo le pomen oznak, ki smo jih uporabili. Znak \ določa, da se samodejno vnese znak, ki sledi poševnici. Torej se bodo vse naše številke avtomatično začele z znakom 0. Oznaka 9 omogoča vnos številke ali presledka. Z zapisano definicijo smo ustvarili vnosno masko oblike 0 \ - - - s skupno dolžino 11 znakov za vnos mobilne številke. Za vnos stacionarne številke uporabimo masko \09\999\-99\-99, ki predpisuje, da so telefonske številke oblike 0 / - - - s skupno dolžino 12 znakov.

V tabeli IZVEDBA (slika 17) smo polju VzorecPoslan določili kratko obliko datuma. To pomeni, da bodo vnešeni datumi prikazani v obliki dd.mm.llll. Določili smo tudi, da ima polje napis »Vzorec poslan dne«. Ostale lastnosti za polje VzorecPoslan smo pustili takšne, kot jih je naredil program Access sam.

Ime polja	Podatkovni tip	Opis
StrID	Besedilo	Dvomesni del mednarodne oznake.
SchID	Besedilo	Trimestni del mednarodne oznake.
SolaID	Besedilo	Nacionalna oznaka šole.
ProgramID	Besedilo	Oznaka učnega programa.
OsebaID	Samoštevilo	Oznaka koordinatorja izvedbe.
VzorecPoslan	Datum/Čas	Datum ko je bil šoli poslan vzorec.
StDijakov	Število	Število vzorčenih dijakov.
ScQPoslan	Datum/Čas	Datum ko je bil poslan šolski vprašalnik.
ScQPrejet	Datum/Čas	Datum ko je šola prejela vprašalnik.
Soglasja	Datum/Čas	Datum ko so bila poslana soglasja staršev.
PorociloPoslano	Datum/Čas	Datum ko je bilo poslano poročilo.
Opombe	Besedilo	Prostor za posebnosti. 255 znakov

**Lastnosti polja**

Splošno | Iskanje

Oblika	Kratka oblika datuma
Vnosna maska	
Napis	Vzorec poslan dne
Privzeta vrednost	
Veljavnostno pravilo	
Veljavnostno besedilo	
Zahtevano	Ne
Indeksirano	Ne
Način IME	Brez nadzora
Stavčni način IME	Brez
Pametne oznake	

Podatkovni tip določa vrsto vrednosti, ki jih lahko uporabi...

Slika 17 Pogled načrta tabele IZVEDBA

Ime polja	Podatkovni tip	Opis
ObmocjeID	Besedilo	Zaporedna številka območja.
Enota	Besedilo	Ime območne enote Zavoda RS za šolstvo.

**Lastnosti polja**

Splošno | Iskanje

Velikost polja	1
Oblika	
Vnosna maska	
Napis	
Privzeta vrednost	
Veljavnostno pravilo	
Veljavnostno besedilo	
Zahtevano	Da
Dovoli ničelno dolžino	Ne
Indeksirano	Da (podvojeni vnosi niso dovoljeni)
Stiskanje Unicode	Ne
Način IME	Brez nadzora
Stavčni način IME	Brez
Pametne oznake	

Ime polja je vključeno s presledki lahko dolgo do 64 znakov. Pomoč o imenih polj...

Slika 18 Pogled načrta tabele OBMOCJE

Ime polja	Podatkovni tip	Opis
StrID	Besedilo	Dvomesni del mednarodne oznake.
SchID	Besedilo	Trimestni del mednarodne oznake.
IzvajalecID	Samoštevilo	Zaporedna številka izvajalca.
IzvedbaID	Število	Številka izvedbe. Privzeta vrednost 1.
Datum	Datum/Čas	Datum izvedbe.
Ura	Datum/Čas	Ura izvedbe.
StrosekIzvedba	Valuta	Honorar za izvajalca.
PrevozenihKM	Število	Število prevoženih kilometrov izvajalca.
Cestnina	Valuta	Strošek cestnine.
Drugo	Valuta	Drugi stroški (vlak, parkirnina itd)
PaketOddan	Datum/Čas	Datum ko je paket oddan izvajalcu.
PaketVrnjen	Datum/Čas	Datum ko je izvajalec vrnil paket.
Opombe	Besedilo	Morebitne opombe. 255 znakov

Lastnosti polja	
Splošno	Iskanje
Oblika	Valuta
Decimalna mesta	Samodejno
Vnosna maska	
Napis	
Privzeta vrednost	5000
Veljavnostno pravilo	
Veljavnostno besedilo	
Zahtevano	Ne
Indeksirano	Ne
Pametne oznake	

Slika 19 Pogled načrta tabele IZVEDBAIZVAJALEC

Tabela OBMOČJE (slika 18) ne vsebuje posebnosti. V tabeli IZVAJALECIZVEDBA (slika 19) smo polju StrosekIzvedba določili privzeto vrednost 5000, polju IzvedbaID pa vrednost 1.

Ime polja	Podatkovni tip	Opis
SolaID	Besedilo	Nacionalna oznaka šole.
ImePolno	Besedilo	Dolgo ime šole, 255 znakov.
ImeKratko	Besedilo	Skrajšano ime šole na 50 znakov.
Naslov	Besedilo	Naslov šole.
PostnaSt	Besedilo	Poštna številka šole.
Seznam15	Da/Ne	Ali je šola poslala seznam 15-letnikov?
ObmocjeID	Besedilo	V katero območje spada šola.
Opombe	Besedilo	Morebitne opombe.

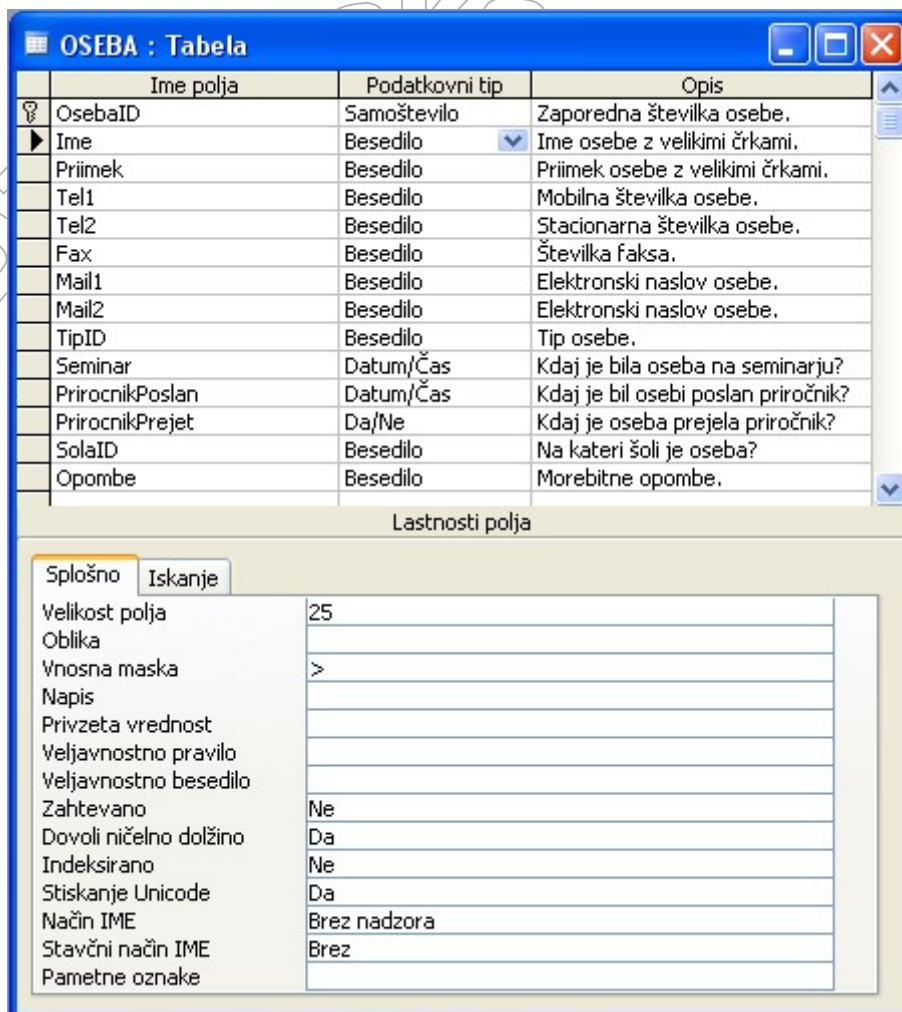
  

Lastnosti polja	
Splošno	
Velikost polja	255
Oblika	
Vnosna maska	
Napis	
Privzeta vrednost	
Veljavnostno pravilo	
Veljavnostno besedilo	
Zahtevano	Ne
Dovolj ničelno dolžino	Da
Indeksirano	Da (podvojeni vnosi dovoljeni)
Stiskanje Unicode	Da
Način IME	Brez nadzora
Stavčni način IME	Brez
Pametne oznake	

Slika 20 Pogled načrta tabele SOLA

V tabeli SOLA (slika 20) smo polje ImePolno indeksirali, saj bomo po njem večkrat iskali šole. Za vnos poštne številke smo določili vnosno masko 0000, kar pomeni da zahtevamo vnos štirih števk.

V tabeli OSEBA (slika 21) smo za polje Ime določili vnosno masko >, kar pomeni da bo vnešeno ime vedno pretvorjeno v velike črke. S tem se bomo izognili različnim oblikam imen, če bi en uporabnik vnašal imena z velikimi črkami, drugi z malimi. Prav tako ne bo potrebno paziti na velike začetnice.

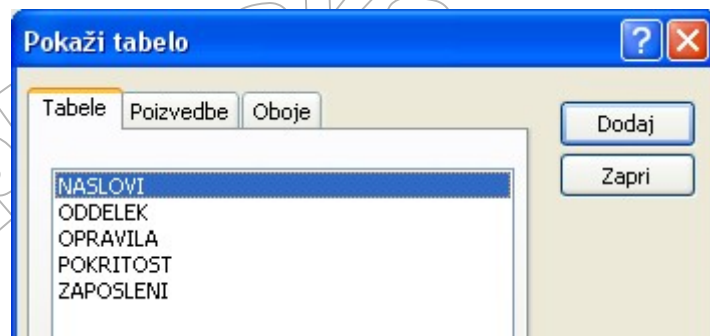


Slika 21 Pogled načrta tabele OSEBA

#### 5.2.4 Vzpostavlanje relacij med tabelami

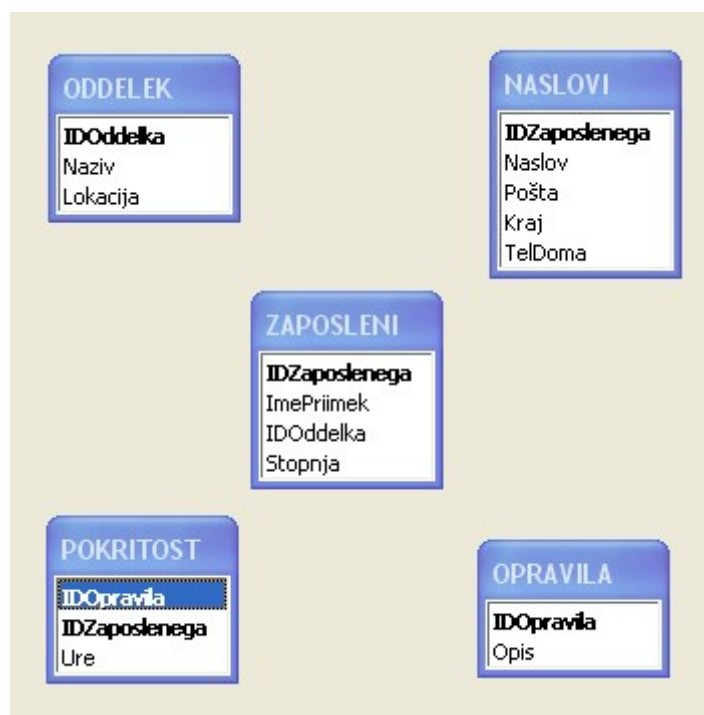
V tem razdelku si bomo ogledali, kako med ustvarjenimi tabelami vzpostavimo relacije in s tem naredimo "pravo" bazo. Na začetku bomo zaradi enostavnosti spet uporabili kar tabele iz poglavja 2.2.1. in na njih razložili osnovne pojme o izdelavi relacij v programu Access. Za razlago na teh primerih smo se odločili, ker bi razlaga na naši načrtovani bazi zahtevala vnos podatkov, česar pa še ne želimo. Tudi za bralca, ki ne pozna projekta PISA, bo razlaga na tem primeru razumljivejša in bo lažje razumel kasnejše relacije v bazi PisaMS06. Res pa je, da relacij za ta izmišljen primer nismo posebej načrtovali. Naš namen ni izdelati še ene baze, ampak na njej le preizkusiti možnosti, ki nam jih ponuja program Access. Tako se ne bomo toliko ozirali na smiselnost povezav, ampak le na lastnosti, ki jih posamezna relacija ima.

Ko izdelamo vse potrebne tabele, v osnovnem oknu programa Access izberemo Orodja/Relacije. Pokaže se okno (slika 22), kjer izberemo tabele, med katerimi želimo ustvariti povezave. Izberemo kar vse tabele in kliknemo na »Dodaj«.



Slika 22 Izbira tabel med katerimi želimo ustvariti relacije.

Ko dodamo tabele, se odpre okno za urejanje relacij z izbranimi tabelami (slika 23).

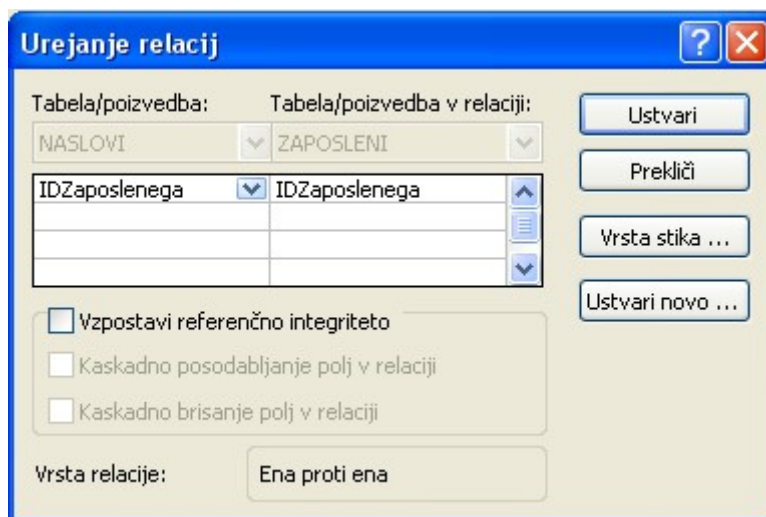


Slika 23 Prikaz izbranih tabel

Med tabelami bomo izdelali različne tipe relacij, da vidimo kakšne možnosti povezovanja imamo v programu Access. Tabele bomo napolnili tudi z nekaj podatki, da bomo opazovali kakšne lastnosti ima posamezna relacija. Sicer si bomo vnašanje podatkov podrobneje pogledali kasneje v poglavju 5.2.5. Začeli bomo z izdelavo najpreprostejše povezave »ena proti ena« in nadaljevali z »ena proti mnogo« in »mного proti mnogo«. Ogleдали si bomo tudi različne stike in referenčno integriteto.

Relacije ustvarimo na zelo preprost način. Z miško primemo polje v prvi tabeli, ki ga želimo povezati in ga povlečemo do polja druge tabele, s katerim ga želimo povezati. V večini primerov je potrebno povleči primarni ključ do tujega ključa. Če je ključ sestavljen iz več polj, s pomočjo tipke Shift ali CTRL označimo ustrezna polja in jih povlečemo do enega od polj povezanega ključa. V naši izmišljeni bazi nimamo nobenega ključa sestavljenega iz več polj, zato tega ne moremo praktično prikazati. Imamo pa tak primer v bazi PisaMS06 in sicer imamo primarni ključ tabele IZVEDBA sestavljen iz dveh polj StID in SchID. Zato si bomo tak primer ogledali kasneje.

Naredimo povezavo ena proti ena med tabelama NASLOVI in ZAPOSLENI. Postavimo se na polje IDZaposlenega v eni od tabel in ga povlečemo do istoimenskega polja v drugi tabeli. Odpre se okno za urejanje relacije (slika 24). V spodnji vrstici okna nam Access pove, da bomo med poljema ustvarili relacijo ena proti ena. V levem zgornjem delu okna vidimo kateri tabeli povezujemo in katera polja. Omenimo še enkrat, da nikakor ni nujno, da se omenjeni polji imenujeta enako. Biti morata le ustreznega tipa, kar smo že razložili v poglavju 2.2.2.

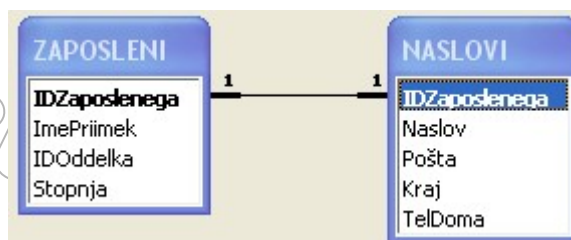


Slika 24 Okno za urejanje relacije

#### 5.2.4.1 Referenčna integriteta

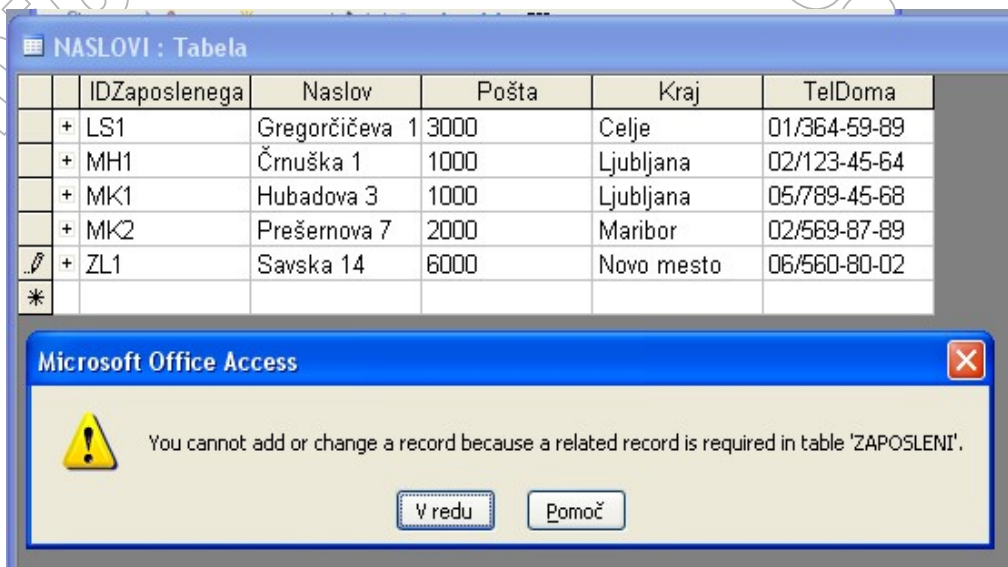
V levem srednjem delu okna (slika 24) imamo možnost vzpostaviti referenčno integriteto. Če vzpostavimo referenčno integriteto, veljajo naslednja pravila:

1. Ob vnosu podatkov v tuji ključ povezane tabele moramo obvezno uporabiti le tiste vrednosti, ki že obstajajo v primarnem ključu primarne tabele. Dovoljeno pa je v tuji ključ vnesti vrednost »nič« (null), kar pomeni, da je zapis nepovezan. Z vrednostjo »nič« označujemo manjkajoče ali neznanе podatke. V našem primeru bi izbira referenčne integritete pomenila, da v tabelo NASLOVI ne moremo vnesti identifikacijske številke zaposlenega, če ne obstaja zapis v tabeli ZAPOSLENI, ki bi že imel tak ID. Preizkusimo. Označimo kvadrček pred »Vzpostavi referenčno integriteto« in potrdimo »ustvari«. Vse ostale nastavitve pustimo take kot so. V oknu se nam med tabelama nariše črta, ki ima na vsaki strani napisano enico (slika 25). To označuje, da gre za relacijo ena proti ena. Če referenčne integritete ne bi izbrali, enic nad črto ne bi bilo. Tako iz grafičnega prikaza relacij lahko hitro razberemo, kje velja referenčna integriteta in kje ne.



Slika 25 Grafični prikaz relacije ena proti ena

Odpremo tabelo NASLOVI in vanjo vnesemo podatke za osebo ZL1, ki v tabeli ZAPOSLENI (slika 27) še ne obstaja. Ko želimo vrstico zapustiti, dobimo sporočilo (slika 26), da ne moremo dodati ali spremeniti zapisa, ker moramo imeti v tabeli ZAPOSLENI ujemajoči zapis - se pravi zaposlenega s primarnim ključem ZL1.



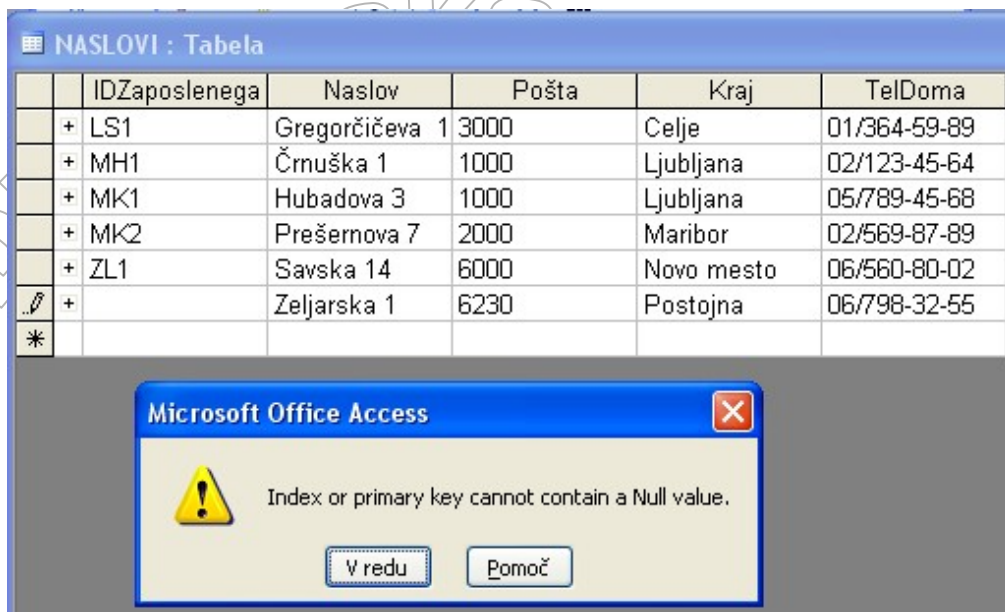
**Slika 26 Opozorilo, da ne moremo dodati novega naslova v tabelo NASLOVI, če ne obstaja ustrezní zapis v tabeli ZAPOSLENI**

IDZaposlenega	ImePriimek	IDOddelka	Stopnja
+ LS1	Lidija Svet	RO	V
+ MH1	Maja Hrust	KO	VII
+ MK1	Miha Kranjc	PO	IV
+ MK2	Mitja Kern	NO	M

**Slika 27 Podatki v tabeli ZAPOSLENI**

Zapis moramo pobrisati (pritisnemo tipko Esc) in najprej vnesti zaposlenega ZL1 v tabelo ZAPOSLENI. To pomeni, da vzpostavitev referenčne integritete zahteva tudi določen vrstni red vnosa podatkov. V našem primeru ne moremo vnesti niti naslova z vrednostjo ključa null, saj je polje IDZaposlenega v tabeli NASLOVI tudi primarni ključ te tabele (slika 28).





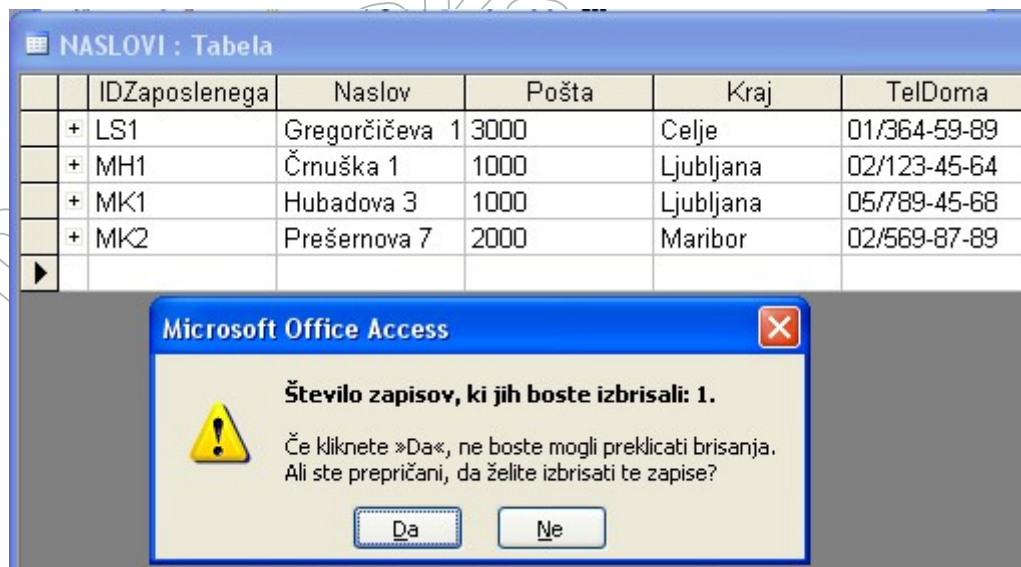
Slika 28 Primarni ključ ne more imeti vrednosti null

2. Iz primarne tabele ne moremo izbrisati zapisa, če v povezani tabeli obstajajo ujemajoči se zapisi. Poskusimo iz tabele ZAPOSLENI izbrisati zaposlenega ZL1, ki smo ga dodali v točki 1.



Slika 29 Opozorilo, da ne moremo pobrisati zapisa v primarni tabeli, ker ima povezan zapis v drugi tabeli

Vrstico označimo in pritisnemo tipko Delete. Dobimo sporočilo, da zapisa ne moremo pobrisati ali spremeniti, ker so v tabeli NASLOVI povezani zapisi (slika 29). Lahko pa pobrišemo naslov v tabeli NASLOVI (slika 30). S tem seveda za tega zaposlenega ne moremo več izvedeti njegovega naslova. Ko torej enkrat vnesemo naslov zaposlenega, zaposlenega ne moremo več izbrisati iz tabele ZAPOSLENI, če prej ne pobrišemo iz tabele NASLOVI njegovega naslova. Torej je tudi v relaciji ena proti ena zelo pomembno, katera tabela je primarna in katera povezana.



Slika 30 Opozorilo ob brisanju zapisa

- Prav tako ne moremo spremeniti vrednosti primarnega ključa, če v povezani tabeli obstajajo povezani zapisi. V tabeli ZAPOSLENI tako ne moremo spremeniti oznake zaposlenega ZL1 v ŽL1. Poskusimo. In res nas program Access opozori, da ne moremo spremeniti ali izbrisati zapisa, ki ima v tabeli NASLOVI povezane zapise (slika 31).



Slika 31 Sporočilo ob nedovoljeni spremembi primarnega ključa

Ko tabelo zapustimo, nas program opozori, da spremembe ne bodo shranjene. To potrdimo in tabela ostane nespremenjena. Če bi torej želeli spremeniti oznako zaposlenega, bi morali najprej v tabeli NASLOVI spremeniti v polju IDZaposlenega ZL1 v ŽL1 (s tem zapis ŽL1 v tabeli ZAPOSLENI ne bi bil povezan z nobenim zapisom v tabeli NASLOVI) in šele potem v tabeli ZAPOSLENI spremeniti oznake zaposlenega ZL1 v ŽL1. Zapisa bi bila ponovno povezana.

Kot vidimo, je referenčna integriteta sistem pravil, ki jih uporablja Microsoft Access, da zagotovi, da so relacije med zapisi veljavne in da po pomoti ne pobrišemo ali spremenimo povezanih podatkov. Poudariti je potrebno, da se spremembe nanašajo le na povezano polje.

Ostala polja zapisa lahko poljubno popravljamo in tega program ne bo zaznal. Tako lahko npr. stopnjo izobrazbe zlahka spremenimo tudi ob vzpostavljeni referenčni integriteti.

Naštajmo pogoje, ki morajo biti izpolnjeni, da bo integriteta delovala:

- Ujemajoče se polje iz primarne tabele je primarni ključ ali pa ima enolični indeks (podvojeni vnosi niso dovoljeni).
- Povezana polja so istega podatkovnega tipa. Obstajata dve izjemi, ki za nas nista pomembni in ju ne bomo podrobneje omenjali.
- Obe tabeli pripadata isti zbirki podatkov. Z nekaj omejitvami je sicer možno povezati tudi tabeli iz različnih baz. A s tem se ne bomo ukvarjali, saj imamo v našem primeru vse tabele v eni podatkovni bazi.

Pri podatkovnih bazah je zelo pomembno, da vanje vnašamo prave podatke. Pri veliki količini podatkov zlahka pride do napake. Zato nam Access ob vzpostavitvi relacij ponuja še dve možnosti. Z njima zmanjšamo možnost vnosa napačnih podatkov. Na sliki 24 vidimo, da imamo z vzpostavitvijo referenčne integritete na izbiro dve dodatni možnosti in sicer »Kaskadno posodabljanje polj v relaciji« in »Kaskadno brisanje polj v relaciji«.

**Kaskadno posodabljanje polj** omogoča spreminjanje primarnih ključev in s tem tudi vseh povezanih zapisov. Vrnimo se torej nazaj in relacijo popravimo tako, da označimo možnost kaskadnega posodabljanja polj. V oknu za urejanje relacij z desnim gumbom kliknemo na relacijo (črta med tabelama) in izberemo »Urejanje relacije« (slika 32).



Slika 32 Spreminjanje relacij

Ponovno se prikaže okno, kot je na sliki 24. Izberemo možnost kaskadnega posodabljanja polj. Ponovno poskusimo popraviti enega od zapisov v tabeli ZAPOSLjeni. Spremenimo MK1 v LK1 (slika 33). Program zapis popravi brez opozorila.

ZAPOSLENI : Tabela				
	IDZaposlenega	ImePriimek	IDOddelka	Stopnja
	+ LS1	Lidija Svet	RO	V
	+ MH1	Maja Hrust	KO	VII
	+ MK1	Miha Kranjc	PO	IV
	+ MK2	Mitja Kern	NO	IV
	+ ZL1	Zana Leban	PO	VI
	+ ŽL1	Živa Leban	PO	VI
*				

Slika 33 Sprememba zapisa MK1 v LK1

Kot smo pričakovali, se je samodejno popravil tudi povezani zapis v tabeli NASLOVI (slika 34).

NASLOVI : Tabela					
	IDZaposlenega	Naslov	Pošta	Kraj	TelDoma
	+ LK1	Hubadova 3	1000	Ljubljana	05/789-45-68
	+ LS1	Gregorčičeva 1	3000	Celje	01/364-59-89
	+ MH1	Črnuška 1	1000	Ljubljana	02/123-45-64
	+ MK2	Prešernova 7	2000	Maribor	02/569-87-89
	+ ŽL1	Savska 14	6000	Novo mesto	06/560-80-02
*					

Slika 34 Popravljen povezan zapis v tabeli NASLOVI

Tretja izbira **kaskadno brisanje polj v relaciji** nam omogoči brisanje primarnih ključev, s tem pa tudi vseh povezanih zapisov. Še enkrat popravimo relacijo tako, da izberemo kaskadno brisanje polj v relaciji. Poskusimo pobrisati enega od zapisov v tabeli ZAPOSLENI. Odločili smo se za brisanje zapisa, ki ima IDZaposlenega enak LS1. Program izpiše opozorilo: »Relacije, ki določajo kaskadno brisanje, bodo pravkar povzročile brisanje 1 zapisov v tej tabeli in v tabelah, povezanih z njo« in nas vpraša, če smo prepričani, da to želimo. Z odgovorom da za vedno pobrišemo izbrane podatke. Če odpremo tabelo NASLOVI vidimo, da je zapis, kjer je bil IDZaposlenega enak LS1, res izbrisan (slika 35).

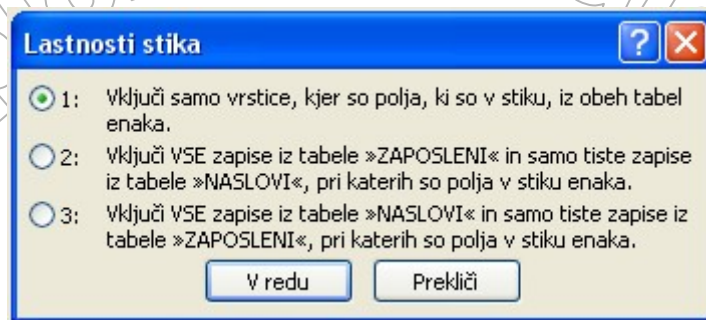
NASLOVI : Tabela					
	IDZaposlenega	Naslov	Pošta	Kraj	TelDoma
	+ MH1	Črnuška 1	1000	Ljubljana	02/123-45-64
	+ MK1	Hubadova 3	1000	Ljubljana	05/789-45-68
	+ MK2	Prešernova 7	2000	Maribor	02/569-87-89
	+ ŽL1	Savska 14	6000	Novo mesto	06/560-80-02
*					

Slika 35 Pobrisan zapis LS1 v povezani tabeli NASLOVI

#### 5.2.4.2 Vrsta stika

Na oknu za urejanje relacij (slika 24) imamo na desni strani izbiro »Vrsta stika«. Ob njeni izbiri se odpre okno (slika 36) s tremi možnostmi. **Stik** je povezava med poljem v eni tabeli ali poizvedbi in poljem enakega podatkovnega tipa v drugi tabeli ali poizvedbi. Stik pove

programu, v kakšni relaciji so zapisi. Odvisno od vrste stika je mogoče zapise, ki se ne ujemajo, vključiti v stik ali pa jih izključiti iz njega.



Slika 36 Lastnosti stika

Najprej opišimo vrste stikov na splošno. V relacijskih podatkovnih bazah poznamo notranji in zunanji stik. Zunanji stik delimo še na levega in desnega. **Notranji stik** poveže zapise z enakimi stičnimi polji v obeh tabelah. Stično polje je polje, ki je v stiku z drugim stičnim poljem. Ponavadi so ta polja primarni in tuji ključi. **Zunanji stik** poveže zapise tako, da vključi vse zapise iz leve ali desne tabele, četudi v drugi tabeli ni ujemajočih se zapisov. Pri levem zunanjem stiku vse zapise prispeva leva (primarna) tabela, se pravi tista s primarnim ključem. Pri desnem zunanjem stiku pa vse zapise prispeva desna tabela.

Poglejmo si primer. Uporabimo Tabelo1 in Tabelo2 z naslednjimi zapisi:

Tabela1	
Oznaka	Ime
A1	Janko
A1	Metka
A2	Andrej
A3	Anton

Tabela 16 Tabela 1

Tabela2	
Štev	ID
1	A1
3	A1
2	A2
6	A2
4	A4
5	A4

Tabela 17 Tabela 2

Kot stični polji bomo uporabili polje Oznaka v prvi tabeli in ID v drugi tabeli. Namenoma smo uporabili polji, ki nista primarna ključa, imata podvojene vrednosti, določene vrednosti so v prvi tabeli in v drugi niso in obratno. Tako smo res zajeli vse možnosti. Določene bi v primeru, če bi bili stični polji primarna ključa in če bi vzpostavili referenčno integriteto, seveda odpadle.

Notranji stik			
Oznaka	Ime	Štev	ID
A1	Janko	1	A1
A1	Janko	3	A1
A1	Metka	1	A1
A1	Metka	3	A1
A2	Andrej	2	A2
A2	Andrej	6	A2

Tabela 18 Rezultat notranjega stika med tabelama

Levi zunanji stik			
Oznaka	Ime	Štev	ID
A1	Janko	1	A1
A1	Janko	3	A1
A1	Metka	1	A1
A1	Metka	3	A1
A2	Andrej	2	A2
A2	Andrej	6	A2
A3	Anton		

Tabela 19 Rezultat levega zunanjega stika med tabelama

Desni zunanji stik			
Oznaka	Ime	Štev	ID
A1	Metka	1	A1
A1	Janko	1	A1
A2	Andrej	2	A2
A1	Metka	3	A1
A1	Janko	3	A1
		4	A4
		5	A4
A2	Andrej	6	A2

Tabela 20 Rezultat desnega zunanjega stika med tabelama

Na sliki 36 prva možnost predstavlja notranji stik, naslednji dve pa levi in desni zunanji stik. Če ne določimo drugače, program Access samodejno izbere prvo možnost, se pravi notranji stik. Praktično so razlike vidne kasneje, pri izvajanju poizvedb. Program bo namreč takrat v rezultat poizvedbe vključil le zapise, ki ustrezajo stiku. Vrnil nam bo torej le ujemajoče zapise, če smo izbrali notranji stik ali pa vse zapise iz ene od tabel in le ujemajoče iz druge glede na to, ali smo izbrali levi ali desni stik. Torej je potrebno že pri ustvarjanju relacij razmisliti, kaj bomo potrebovali pri kasnejših poizvedbah.

Za relacijo med tabelama ZAPOSLENI in NASLOVI bomo izbrali notranji stik. S tem bomo kasneje v poizvedbi dobili v rezultatu le zapise o zaposlenih, ki imajo v tabeli NASLOVI pripadajoči naslov.

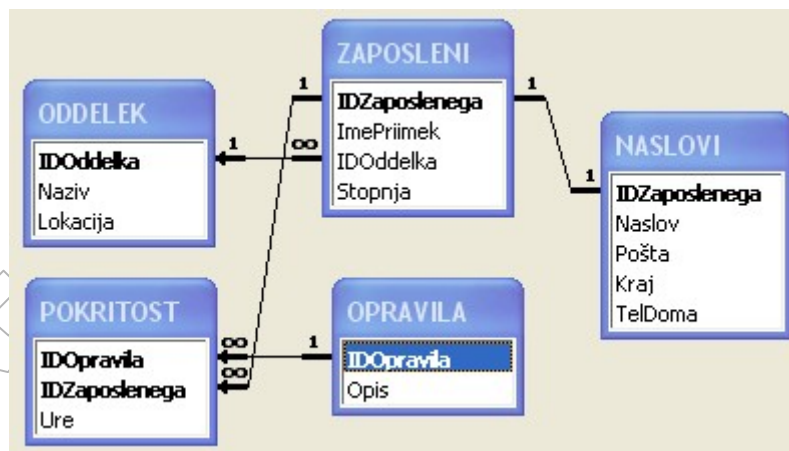
#### 5.2.4.3 Vzpostavlanje relacij

Ustvarimo še relacije med ostalimi tabelami in razmislimo, katere lastnosti jim bomo določili. Med tabelama ODDELEK in ZAPOSLENI imamo relacijo ena proti mnogo.

Izberemo referenčno integriteto z možnim kaskadnim posodabljanjem in brisanjem polj in desnim stikom. To pomeni, da v relacijo vključimo vse zapise iz desne tabele ZAPOSLENI in le ujemajoče zapise iz tabele ODDELEK. Ujemajoč zapis pomeni zapis v levi tabeli, ki ima enako vrednost povezanega polja kot polje v desni tabeli. Kot vidimo v prejšnjem poglavju (tabela 20) zapisa A3 ni, ker ne se ne ujema z nobenim zapisom v desni tabeli. Zapis A4 iz desne tabele je viden kljub temu, da leva tabela nima ujemajočega zapisa. Zato dobimo na levi vrednosti null.

Stik označuje puščica na sliki 37, ki vedno kaže na tabelo, ki prispeva le ustrezne zapise. V našem primeru bi dobili vse zapise iz tabele ZAPOSLENI tudi, če v tabeli ODDELEK pripadajoči oddelek ne obstaja. V rezultatu (združeni tabeli) pač ne bi bilo podatkov iz tabele ODDELEK. In tukaj se moramo vrniti nekoliko nazaj. Prej smo rekli, da referenčna integriteta pomeni, da v tuj ključ ne moremo vnesti vrednosti, ki ne obstaja v primarni tabeli. To pa je v protislovju z opisom desnega zunanjšega stika. Če smo izbrali referenčno integriteto, bomo pri desnem stiku torej navidezno dobili notranji stik. Navidezno zato, ker bo enak le rezultat, pot do tega rezultata pa ne bo enaka. Pri notranjem stiku program pogleda zapis v primarni tabeli in potem preveri, če je v povezani tabeli ujemajoči se zapis. Pri desnem stiku pa bo vzel vse zapise iz povezane tabele in ujemajoče iz primarne tabele. To bo prineslo enak rezultat, saj se v primarni tabeli zaradi referenčne integritete nahajajo vsi zapisi, ki so v povezani tabeli. Na sliki 37 sta nad povezavo tokrat oznaki 1 in  $\infty$ , ki označujeta referenčno integriteto v relaciji ena proti mnogo. Referenčna integriteta bo poskrbela, da ne bomo imeli v bazi zaposlenega na oddelku, ki ne obstaja. Kaskadno brisanje in posodabljanje nam bo omogočalo premeščanje delavcev na druge oddelke in izbris oddelka ob morebitnem zaprtju le tega. V primeru, da bi ob zaprtju oddelka odpustili vse zaposlene, bi enostavno pobrisali oddelek in kaskadno brisanje bi za nas izbrisalo iz baze tudi vse zaposlene na tem oddelku. Seveda pa lahko nekatere prej premestimo na drug oddelek tako, da spremenimo v tabeli ZAPOSLENI polje *IDOddelka* v ustrezno novo vrednost.

Povezati moramo še tabeli ZAPOSLENI in OPRAVILA s pomočjo stične tabele POKRITOST. Obe tabeli moramo povezati s stično tabelo z relacijo ena proti mnogo. Izbrali smo referenčno integriteto s kaskadnim brisanjem in posodabljanjem ter levi stik. To je razvidno iz puščic na sliki 37, ki kažeta na stično tabelo. To pomeni, da v relacijo vključimo vse zaposlene in vsa opravila tudi v primeru, da še nimajo pokritosti. Zanimalo nas bo namreč, kateri zaposleni še nima dela in katero opravilo še ni dodeljeno.



Slika 37 Prikaz vzpostavljenih relacij

Poglejmo si lastnosti referenčne integritete še na nekaj primerih. V tabelo POKRITOST smo poskusili v polje IDzaposlenega vnesti LS1 (slika 38), vendar nam je program javil sporočilo (slika 39), da vrednosti ne moremo vnesti, ker v tabeli ZAPOSLENI ta delavec ne obstaja.

	IDOpravila	IDZaposlenega	Ure
	1	MK2	1
	3	MH1	30
	4	MK1	16
	3	LS1	4
*			0

Slika 38 Poskus vnosa 'nezaposlenega'



Slika 39 Opozorilo da 'nezaposlenega' ne moremo vnesti

V tabelo POKRITOST (slika 40) vnesimo še nekaj podatkov in si pogledjmo, kaj se zgodi, če spremenimo ali izbrisemo zapis v tabeli ZAPOSLENI ali OPRAVILA.

	IDOpravila	IDZaposlenega	Ure
▶	1	MK1	34
	1	MK2	1
	2	ZL1	13
	2	ŽL1	3
	3	MH1	30
	3	ŽL1	4
	4	MK1	16
	4	ZL1	5
*			0

Slika 40 Podatki v tabeli POKRITOST

Poskusimo v tabeli ZAPOSLENI spremeniti MK1 v MK15, ter si oglejmo tabelo POKRITOST ponovno. Na sliki 41 vidimo, da je program samodejno, brez opozorila, popravil vse vrednosti MK1 v MK15.

Pobrišimo v tabeli OPRAVILA opravilo številka 1. Na sliki 42 vidimo, da smo v tabeli POKRITOST izgubili vse zapise z opravilom 1.



POKRITOST : Tabela			
	IDOpravila	IDZaposlenega	Ure
▶ 1		MK15	34
1		MK2	1
2		ZL1	13
2		ŽL1	3
3		MH1	30
3		ŽL1	4
4		MK15	16
4		ZL1	5
*			0

Slika 41 Popravljen oznaka zaposlenega MK1 v MK15

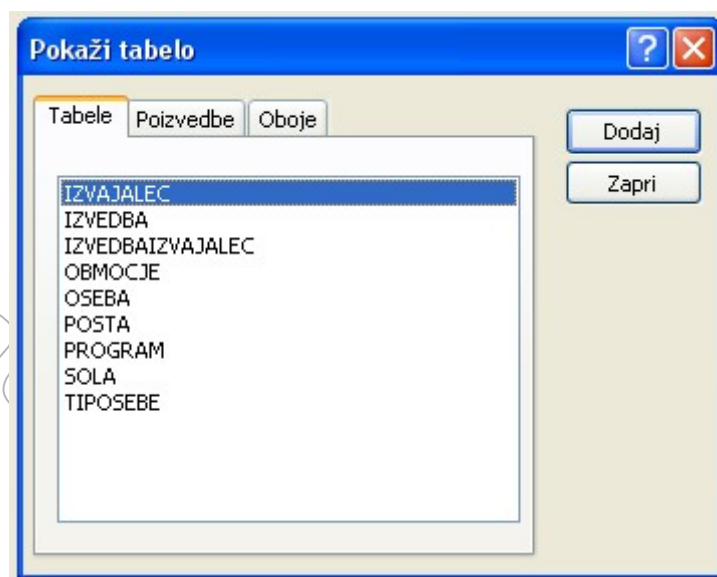
POKRITOST : Tabela			
	IDOpravila	IDZaposlenega	Ure
▶ 2		ZL1	13
2		ŽL1	3
3		MH1	30
3		ŽL1	4
4		MK15	16
4		ZL1	5
*			0

Slika 42 Izbrisano opravilo 1 je povzročilo izbris pripadajočih pokritosti

#### 5.2.4.4 Vzpostavlanje relacij v bazi PisaMS06

Sedaj ko natančno vemo, kaj nam program ponuja, se lotimo izdelave relacij v bazi PisaMS06.

Izberemo vse tabele (slika 43), saj želimo ustvariti vse povezave, ki smo jih načrtovali v poglavju 5.1.5. In v teh povezavah nastopajo vse tabele.



Slika 43 Izbira tabel med katerimi želimo ustvariti relacije.

Pri ustvarjanju povezav si bomo pomagali z modelom ER na sliki 9. Vse relacije v naši bazi so tipa ena proti mnogo. Izjema je bila le ena relacija tipa mnogo proti mnogo, ki pa smo jo tudi pretvorili v dve relaciji ena proti mnogo. V naši bazi PisaMS06 nimamo nobene relacije tipa ena proti ena.

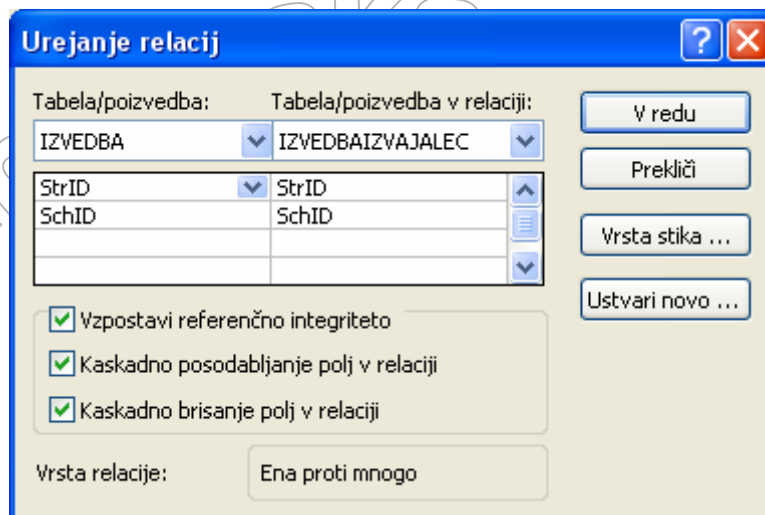
Pričeli bomo z izdelavo povezave med tabelo TIPOSEBE in OSEBA. Ker uporabniki programa Access pogosto napačno menijo, da se morajo imena stičnih polj (polji, ki ju povezujemo) ujemati, omenimo še enkrat (že tretjič), da to nikakor ni nujno. Kot smo razložili že v poglavju 2.2.2., morata biti le ustreznega tipa. Relacija med tabelama nam bo omogočala združevanje informacij iz obeh tabel. Denimo, da nas bo zanimalo, katere osebe so ravnatelji, katere koordinatorji in katere oboje. To bomo izvedeli s pomočjo poizvedbe tako, da bomo programu ukazali, naj nam vrne vse osebe, ki imajo vrednost tujega ključa TipID enako določenemu primarnemu ključu iz tabele TIPOSEBE. Vzpostavili bomo referenčno integriteto brez možnosti kaskadnega posodabljanja in brisanja ter uporabili privzeti notranji stik. Tipe imamo samo tri in predpostavljamo, da se ne bodo spreminjali. Zato bomo preprečili popravljanje in brisanje tipov iz baze.

Povezave z enakimi lastnostmi bomo določili tudi med tabelami OBMOCJE-SOLA, POSTA-SOLA in POSTA-IZVAJALEC ter PROGRAM-IZVEDBA.

V relaciji med tabelama SOLA in OSEBA bomo uporabili levi stik, da nam bo kasnejša poizvedba vrnila vse šole, tudi tiste, pri katerih v tabeli OSEBA še ne bomo imeli vnešene kakšne osebe. S tem bomo hitro videli, na kateri šoli nam manjkajo podatki o kontaktni osebi. Podatkov o šolah ne bomo spreminjali ali brisali iz baze, zato tudi tu ne bomo izbrali kaskadnega posodabljanja in brisanja. Relacije z enakimi lastnostmi bomo vzpostavili tudi med pari tabel OSEBA-IZVEDBA ter SOLA-IZVEDBA. Tudi tukaj nam ustreza levi stik. Pri poizvedbah bomo želeli dobiti vse osebe in vse šole tudi, če še nimajo vnešene izvedbe. Pogosto nas bo to ravno najbolj zanimalo.

Preostane nam le še povezava tabel IZVEDBA in IZVAJALEC s stično tabelo IZVEDBAIZVAJALEC. Tu bomo za razliko od ostalih povezav izbrali možnost kaskadnega posodabljanja in brisanja polj. Izvedbe so mednarodno določene in se iz cikla v cikel spreminjajo. Včasih se tudi izkaže, da je zaradi napake v podatkih, ki jih nacionalnemu centru posreduje šola kakšna izvedba z različno identifikacijsko številko pravzaprav ista (zajema iste učence). Zato bo potrebno odvečno izvedbo pobrisati iz baze. Prav tako nam ustreza levi zunanji stik, ki nam bo omogočal pridobiti iz baze zapise o vseh izvedbah tudi, če katera še ne bo dodeljena izvajalcu.

V poglavju 5.2.4. smo obljubili tudi prikaz ustvarjanja relacije, če je primarni ključ sestavljen iz več polj. S pomočjo tipke Ctrl smo označili polji StrID in SchID v tabeli IZVEDBA in z miško potegnili do polja StrID v tabeli IZVEDBAIZVAJALEC. Lahko bi potegnili tudi do polja SchID. Dobili smo okno za urejanje relacij (slika 44), kjer smo označili v prejšnjem odstavku opisane lastnosti. Na sliki 45 vidimo, da se zaradi izbranega levega stika med poljema ustvarita dve puščici usmerjeni proti tabeli IZVEDBAIZVAJALEC.

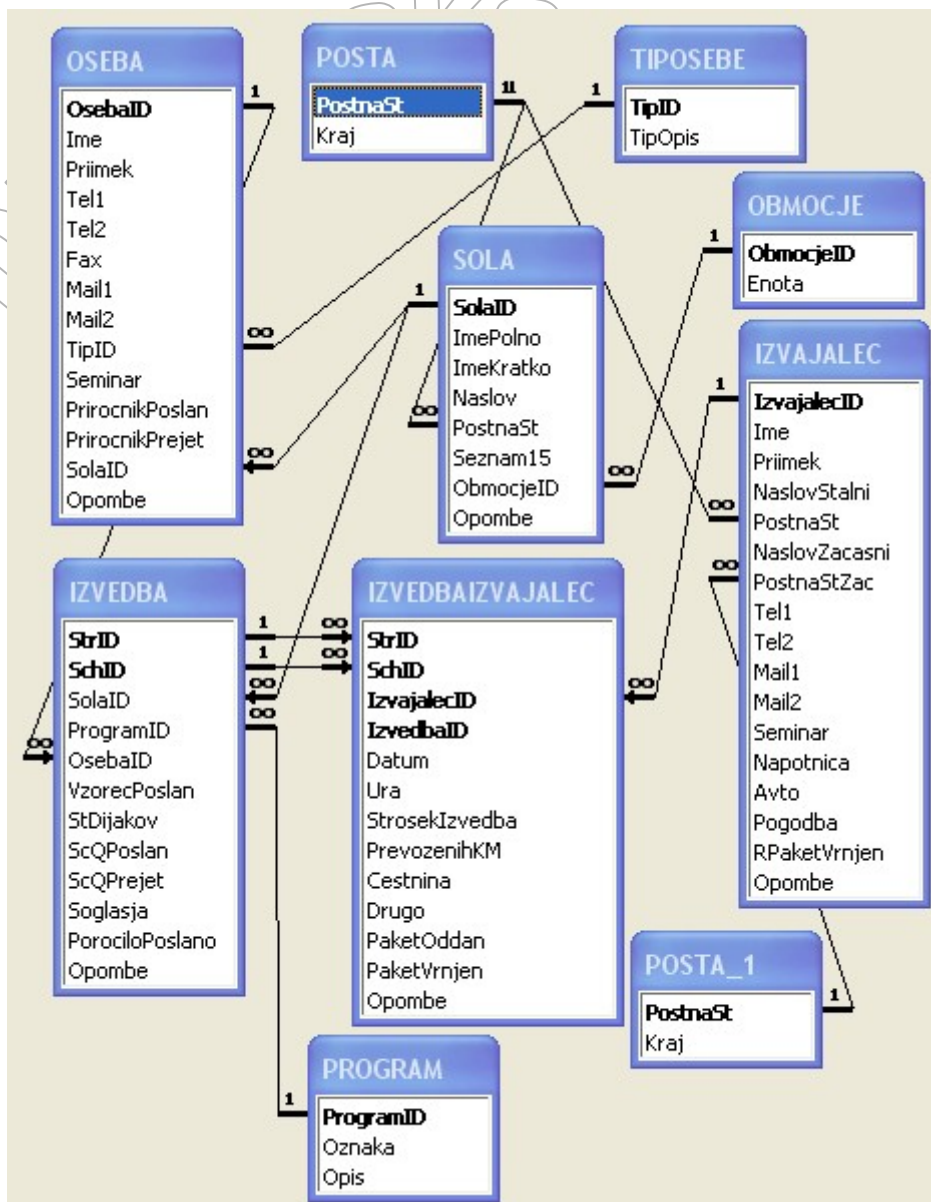


Slika 44 Izbrane lastnosti relacije IZVEDBA - IZVEDBAIZVAJALEC

Relacija IZVAJALEC-IZVEDBAIZVAJALEC bo popravljanje in brisanje zapisov onemogočala. Torej ne bomo izbrali kaskadnega posodabljanje in brisanja. Tudi tukaj bomo uporabili levi stik, da bomo kasneje v poizvedbi dobili zapise vseh izvajalcev tudi, če ne bo pripadajočega zapisa v tabeli IZVEDBAIZVAJALEC. Želimo torej dobiti vse zapise iz tabel IZVAJALEC in IZVEDBA ter le ustrezne zapise iz tabele IZVEDBAIZVAJALEC. Tako bomo hitro opazili, če kakšna izvedba ni nikomur dodeljena ali če je izvajalec brez dodeljenih izvedb.

Realizirali smo vse načrtovane povezave in jih podrobneje opisali. Na sliki 45 jih prikažimo še grafično. Sproti smo že razložili, kaj pomenijo puščice in ostale oznake ob stičnih črtah. Na sliki opazimo tabelo POSTA\_1, ki je do sedaj še nismo omenili. Tabelo doda program Access sam, če povežemo en primarni ključ z dvema tujim ključema, ki sta v isti tabeli. Mi smo povezali primarni ključ PostnaSt s tujima ključema PostnaSt in PostnaStZac, ki se nahajata v tabeli IZVAJALEC. Dejansko se tabela v bazi ne podvoji, ampak se prikaže le vizualno za boljšo preglednost grafičnega prikaza relacij.

Ustvarjene relacije je možno popravljati tudi kasneje, vendar tvegamo izgubo podatkov.



Slika 45 Grafični prikaz relacij in tabel realiziranih v programu Access

### 5.2.5 Vnos podatkov v tabele

Rezultat prejšnjih poglavij je prazna baza PisaMS06. Bazo je potrebno napolniti s podatki. V tem poglavju si bomo ogledali, kako to storimo. V tabele z osebnimi podatki bomo vnesli izmišljene podatke, saj gre le za prikaz izdelave baze in sami podatki niso pomembni. Ker si bomo podatke izmislili, bo obseg podatkov manjši, kot bo v bazi s pravimi podatki. Baza s pravimi podatki bo narejena šele za dejansko uporabo skupaj z uporabniki. Vnešene podatke bomo potrebovali pri izdelavi poizvedb, obrazcev ter poročil.

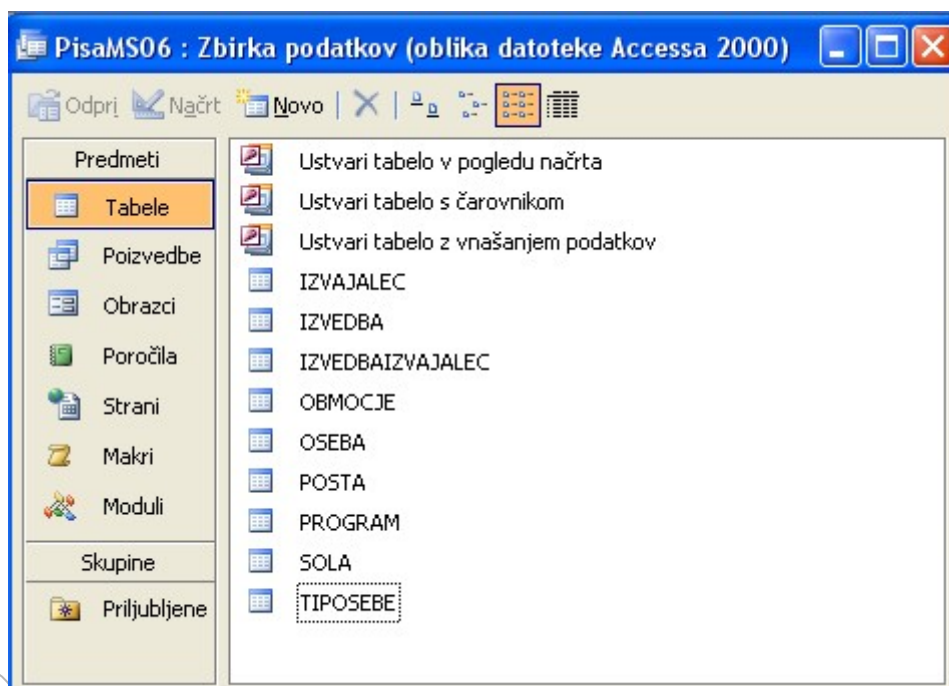
Zaradi referenčne integritete se moramo vnašanja podatkov lotiti sistematično. Paziti moramo namreč, da podatke vnesemo najprej v primarno tabelo in šele potem v povezano tabelo. Naredimo si vrstni red vnašanja podatkov v našo bazo. Po vrsti naštejmo tabele, kot bomo vanje vnašali podatke: TIPOSEBE, PROGRAM, OBMOČJE, POSTA, SOLA, OSEBA, IZVEDBA, IZVAJALEC, IZVEDBAIZVAJALEC. Pri določanju vrstnega reda smo si pomagali s sliko 45. Pri tem smo upoštevali, da je potrebno vedno napolniti vse primarne tabele pred tabelo, ki je z njo povezana. Tako moramo pred tabelo IZVEDBA

napolniti tabele SOLA, OSEBA in PROGRAM. Tabela IZVEDBA namreč vsebuje tuje ključe SolaID, OsebaID in ProgramID, ki so povezani s prej naštetimi tabelami. Tabela IZVEDBA mora biti napolnjena pred tabelo IZVEDBAIZVAJALEC itd.

Pomembno je tudi, da strukturo baze dokončno določimo pred samim vnosom podatkov. Naknadno spreminjanje strukture (dodajanje polj, spreminjanje tipov polj, ...) pogosto vodi do izgube že vnešenih podatkov.

Program Access omogoča vnos podatkov na tri različne načine. Prvi je vnos podatkov neposredno v tabelo. Drugi način je uvoz podatkov iz zunanjih virov. Tretji način je vnos podatkov preko obrazca, kar bomo podrobneje spoznali v poglavju 5.4.

Za vnos podatkov neposredno v tabelo moramo tabelo najprej odpreti. To storimo tako, da na osnovnem oknu (slika 46) dvakrat kliknemo na izbrano tabelo. Zatem v tabelo vnesemo podatke (slika 47), shranimo in tabelo zapremo. Podatke vnašamo tako da se postavimo v prvo polje vrstice v tabeli, vpišemo podatek in se s tipko Enter pomaknemo v naslednje polje v vrstici. Ko pridemo na konec vrstice, nas tipka Enter prestavi v prvo polje naslednje vrstice. Po poljih se lahko premikamo tudi s smernimi tipkami. Na enak način odpremo tudi preostale tabele in vanje vnesemo podatke. Na slikah 47, 48 in 49 vidimo prve tri tabele z vnešenimi podatki.



Slika 46 osnovno okno s tabelami

TIPOSEBE : Tabela		
	TipID	TipOpis
+ 1		Ravnatelj
+ 2		Koordinator
+ 3		RavnateljKoordinator
*		

Slika 47 Tabela TIPOSEBE z vnešenimi podatki.

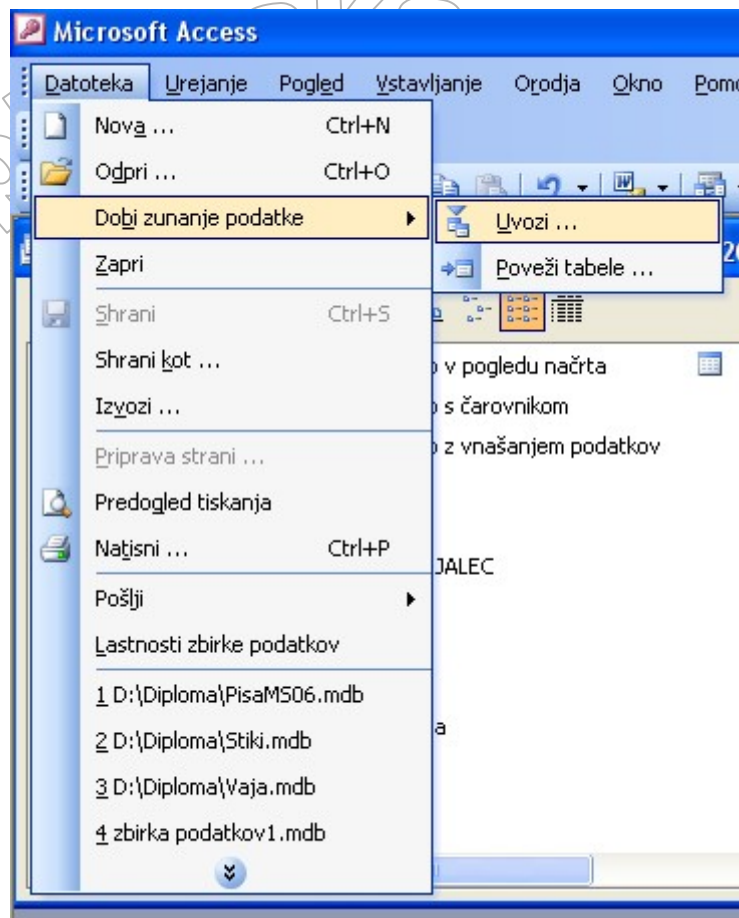
PROGRAM : Tabela			
	ProgramID	Oznaka	Opis
	+ 1	OŠ	Program osnovne šole
	+ 2	NPI	Programi nižjega poklicnega izobraževanja
	+ 3	SPI	Programi srednjega poklicnega izobraževanja
	+ 4	STSI	Programi srednjega strokovnega in tehničnega izobraževanja
	+ 5	GIMS	Program strokovne gimnazije
	+ 6	GIMG	Program splošne in klasične gimnazije
*			

Slika 48 Tabela PROGRAM z vnešenimi podatki

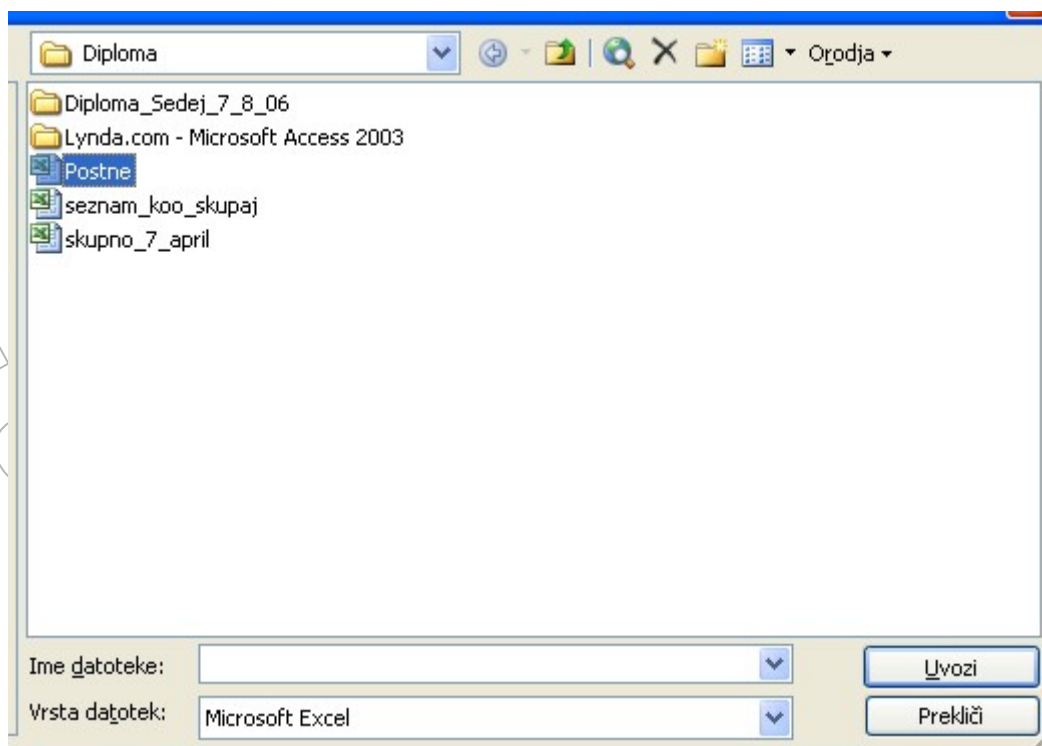
OBMOCJE : Tabela		
	ObmocjeID	Enota
	+ 1	Ljubljana
	+ 2	Maribor
	+ 3	Koper
	+ 4	Nova Gorica
	+ 5	Novo mesto
	+ 6	Murska Sobota
	+ 7	Celje
	+ 8	Kranj
	+ 9	Slovenj Gradec
*		

Slika 49 Tabela OBMOCJE z vnešenimi podatki

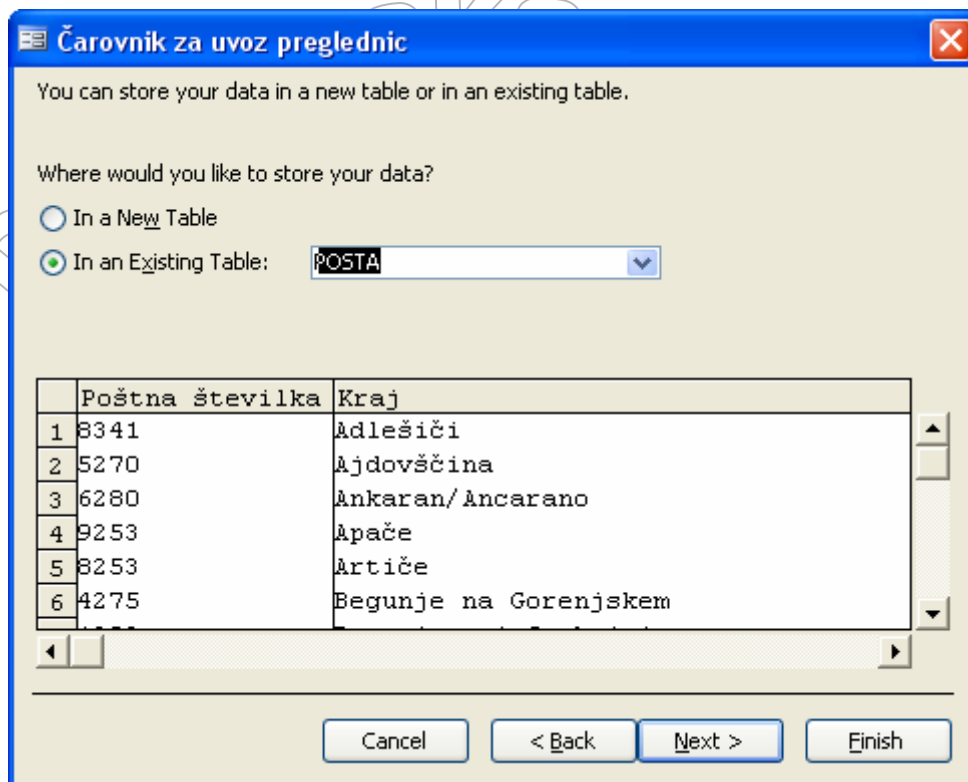
Tabela POSTA vsebuje večjo količino podatkov, zato bi bilo ročno vnašanje podatkov neposredno v tabelo preveč dolgotrajno. Zato bomo pri polnjenju te tabele uporabili drug način vnosa podatkov in sicer pridobivanje podatkov iz drugih virov. Preglednico poštne številke najdemo na spletu v obliki Excelove preglednice. Uvozimo jo v našo tabelo POSTA. Za uvoz podatkov izberemo ukaz, kot ga kaže slika 50. Odpre se okno, v katerem si izberemo vir podatkov – v našem primeru poiščemo datoteko s preglednico poštne številke (slika 51) in izberemo Uvozi. Zatem nas čarovnik vodi skozi uvoz podatkov. V primeru, ko so podatki v Excelovi preglednici, moramo najprej izbrati, ali so podatki na delovnem listu ali v z imenom določenem območju. V našem primeru so številke na območju z imenom Postne. V naslednjem koraku nas čarovnik vpraša, če podatki v prvi vrstici vsebujejo imena stolpcev. V tretjem koraku določimo tabelo, kamor naj se podatki uvozijo (slika 52). Mi si seveda izberemo našo tabelo POSTA. Po potrditvi nas vpraša, če nam po zaključenem uvozu podatke analizira. V našem primeru smo izbrali ne. Po končanem uvozu nas program obvesti o uspehu pri uvozu podatkov.



Slika 50 Ukaz za uvoz podatkov



Slika 51 Okno kjer izberemo vir podatkov



Slika 52 Okno v katerem izberemo tabelo, v katero bomo podatke uvozili

Del napolnjene tabele POSTA vidimo na sliki 53.

	Poštna številka	Kraj
▶ +	1000	Ljubljana
+	1111	Ljubljana - Vič
+	1210	Ljubljana Šentvid
+	1211	Ljubljana - Šmartno
+	1215	Medvode
+	1216	Smlednik
+	1217	Vodice
+	1218	Komenda
+	1219	Laze v Tuhinju
+	1221	Motnik
+	1222	Trojane
+	1223	Blagovica
+	1225	Lukovica
+	1230	Domžale
+	1231	Ljubljana Črnuče
+	1233	Dob
+	1234	Mengeš

Slika 53 Del tabele POSTA z vnešenimi podatki

Pri vnosu podatkov v tabelo SOLA (slika 54) bomo zopet izbrali kar ročen vnos podatkov neposredno v tabelo.



LA : Tabela			
SolaID	ImePolno	ImeKratko	Naslov
101	Biotehniški izobraževalni center Ljubljana - Gim	BIC Ljubljana - Gim.	Cesta v Mestni log 47
102	Biotehniški izobraževalni center Ljubljana - Živi	BIC Ljubljana - Živils	Ižanska c. 10
103	Ekonomška šola Ljubljana		Prešernova 6
104	Gimnazija Bežigrad		Peričeva 4
105	Gimnazija EURO šola Ljubljana		Litistrojska c. 40
106	Gimnazija in ekonomska srednja šola Trbovlje		Gimnazijska 10
107	Gimnazija Jožeta Plečnika Ljubljana		Šubičeva 1

Slika 54 Del tabele SOLA z vnešenimi podatki

Pri vnosu podatkov v tabelo OSEBA ugotovimo, da za polje OsebaID izbrani podatkovni tip Samoštevilico ne bo primeren. Če namreč določen zapis pobrišemo, s tem izgubimo pripadajoči primarni ključ. Tako bi lahko imeli polno vrzeli. Na primer zapis z vrednostjo polja OsebaID 1 bi vseboval podatke o prvi osebi, podatki o drugi pa bi se lahko nahajali denimo v zapisu z vrednostjo polja OsebaID enako 5, itd. Do tega bi prišlo, če smo se pri vnosu večkrat zmotili in zapis pobrisali. Zato bomo tip Samoštevilico zamenjali z tipom Besedilo dolžine 3. To zadošča za 999 oseb, česar pri naši količini podatkov zagotovo ne bomo presegle. Sicer gre bolj za "lepotni" popravek, a z njim bomo ilustrirali, zakaj je dobro, da bi razmislili o ustreznosti strukture tabele podatkov opravili že prej.

Za omenjeni popravek je potrebno namreč najprej odstraniti vse relacije, ki jih ima polje OsebaID z ostalimi polji. Podatkovni tip moramo spremeniti tudi vsem poljem, ki so bila povezana s poljem OsebaID. To je v našem primeru na srečo le polje OsebaID v tabeli IZVEDBA (slika 56). V pogledu načrta naredimo želene popravke in ponovno vzpostavimo relacije. Vnešene podatke vidimo na sliki 55.

OSEBA : Tabela					
	OsebaID	Ime	Priimek	Mobi	Telefon
▶ + 1		MAJA	HRUST	031/563-675	01/569-34-76
+ 2		MIHA	KRANJC	041/049-456	01/589-30-56
+ 3		ŽIGA	MAVER ZAJC	051/593-666	01/637-83-74
+ 4		JANJA	ŠMID	041/465-777	01/636-74-57
*					

Slika 55 Del tabele OSEBA z vnešenimi podatki

IZVEDBA : Tabela								
	StrID	SchID	ID šole	Učni program	OsebaID	Vzorec poslar	StDijakov	ScQPoslan
+ 01	007	101	6			10.3.2006	25	15.4.2006
+ 02	023	106	5		5	15.3.2006	24	13.4.2006
+ 03	015	103	4			15.3.2006	25	13.4.2006
+ 04	006	102	3		1	11.3.2006	20	15.4.2006
+ 05	004	102	2		1	10.3.2006	25	15.4.2006
							25	

Slika 56 Del tabele IZVEDBA z vnešenimi podatki

Enak razlog kot pri tabeli OSEBA nas vodi, da tudi v tabeli IZVAJALEC podatkovni tip polja IzvajalecID spremenimo iz tipa Samoštevilico v tip Besedilo dolžine 3. Del tabele z (izmišljenimi) podatki vidimo na sliki 57.

IZVAJALEC : Tabela							
	IzvajalecID	Ime	Priimek	Stalni naslov	Poštna	Začasni naslov	Poštna
+ 1		LEA	MIKUŽ	Pod Hrasti 3	3254	Kajuhova 10	1000
+ 2		PETRA	ČUK	Clevelandska 5	1000		
+ 3		HELENA	KEBER	Samova ul. 35	2000		
+ 4		MITJA	LAH	Obala 19	6000		
+ 5		ŽAN	JESIH	Dunajska c. 34	1000		
+ 6		JANA	SAVIČ	Slovenska c. 89	1000		
*							

Slika 57 Del tabele IZVAJALEC z vnešenimi podatki

IZVEDBAIZVAJALEC : Tabela							
	StrID	SchID	IzvajalecID	IzvedbaID	Datum	Ura	Stroseklzvedba
	01	007	1	1	21.3.2006	10:00	5.000,00 SIT
	01	007	1	2	30.3.2006	9:00	5.000,00 SIT
	02	023	4	1	4.4.2006	8:15	5.000,00 SIT
	03	015	1	1	1.4.2006	14:00	5.000,00 SIT
	04	006	5	1	17.3.2006	8:00	5.000,00 SIT
	05	004	6	1	23.3.2006	10:20	4.000,00 SIT
				1			5.000,00 SIT

Slika 58 Del tabele IZVEDBAIZVAJALEC z vnešenimi podatki

The screenshot displays the Microsoft Access interface with several tables open. The 'IZVAJALEC' table is the primary focus, showing a list of contractors with their IDs, names, surnames, addresses, and postal codes. Other visible tables include 'TIPOSEBE' (roles), 'OSEBA' (persons), and 'POSTA' (postal codes). The 'IZVEDBA' table shows execution records for these contractors, including dates, times, and costs.

Slika 59 Istočasno odprte tabele med vnosom podatkov

Pri vnašanju podatkov v tabele pogosto potrebujemo pregled nad tem, katere podatke smemo vnesti (tiste, ki so v primarni tabeli). Zato bi radi videli vsebino dveh (ali več) tabel hkrati.

To v programu Access ne predstavlja težav, saj si lahko hkrati odpremo več tabel in jih z ukazom »Okno > Drugo pod drugim ali Drugo ob drugem« razporedimo čez cel zaslon. Primer take razporeditve vidimo na sliki 59. Seveda je pri tem zelo dobro, da je naš zaslon kar se da velik.

V vse tabele smo vnesli nekaj podatkov, kar bo zadoščalo za izdelavo poizvedb, obrazcev in poročil. To bomo opisali v prihodnjih poglavjih.

### 5.3 Izdelava poizvedb

V tem poglavju si bomo ogledali, kako lahko iz baze pridobimo odgovore na vprašanja, ki smo jih navedli v poglavju 5.1.1. S pomočjo poizvedb iz množice podatkov, ki jih imamo spravljene v bazi podatkov, pridobivamo želene informacije. Poizvedba je le pogled na informacije shranjene v osnovnih tabelah. Torej to niso nove tabele v bazi, temveč le našeti kriteriji in pogoji za prikaz podatkov iz osnovnih tabel. Access nam prikaže tiste podatke in kombinacije podatkov, ki ustrezajo postavljenim kriterijem in pogojem. Prikazani podatki se ne shranijo, temveč Access shrani le poizvedbo (kriterije, pogoje) in ob ponovni izvedbi poizvedbe podatke ponovno črpa iz baze. Prikazanemu podatkom pravimo dinamični set zapisov (dynaset), ki se zgradi vedno, ko poženemo poizvedbo. Ko jo zapremo, se ta skupek zapisov (ki so rezultat poizvedbe) izbriše iz pomnilnika.

Poizvedbe v programu Access lahko naredimo na dva načina. Prvi je grafični in se imenuje Poizvedba preko primera (Query By Example - QBE). Drugi pa je z uporabo programskega jezika SQL. Pri tem uporabimo v Access vgrajen urejevalnik za jezik SQL. V slednjem bomo poizvedbe izdelovali tudi mi. Na ta način pridobljeno znanje bo uporabno tudi v drugih programskih okoljih, saj je, kot smo že omenili, jezik SQL standardni jezik za izvedbo poizvedb.

Poizvedbe bomo razdelili na enostavne in sestavljene. Prve izvajamo po posameznih tabelah, druge pa po več tabelah hkrati.

#### 5.3.1 Enostavne poizvedbe

Enostavne poizvedbe izvajamo znotraj podatkov ene tabele. Bistveno pri poizvedbi je, da vemo katere podatke želimo prikazati in kakšni so kriteriji za prikaz teh podatkov.

Za izdelavo poizvedbe si moramo odpreti urejevalnik za SQL. V osnovnem oknu naše baze v levem stolpcu kliknemo na Poizvedbe in izberemo ukaz Novo. Kot kaže slika 60, se odpre okno, kjer izberemo Pogled načrta. Odpre se okno za izbiro tabel pri izdelavi poizvedb v načinu QBE (slika 61). Ker poizvedb ne bomo izdelovali na ta način, izberemo Zapri. Preselimo se v ukazno vrstico in izberemo »Pogled > SQL Pogled«. Odpre se urejevalnik za jezik SQL (slika 62).

V oknu vidimo že vnešen osnovni ukaz poizvedovanja po podatkih – ukaz »SELECT;«. Temu sledi poizvedba, sestavljena iz treh delov. Najprej povemo, **kaj** hočemo dobiti, nato v **kateri tabeli se nahajajo podatki**, ki jih potrebujemo za poizvedbo. V tretjem delu pa povemo, kakšne so omejitve oziroma **pogoji** za prikaz željenih podatkov. Če pogojev ni, tretji del spustimo.

Ukazu v SQL za poizvedbo pravimo tudi stavek SELECT in je videti takole:

```
SELECT atributi, ki jih želimo dobiti  
FROM tabela, v kateri so podatki
```

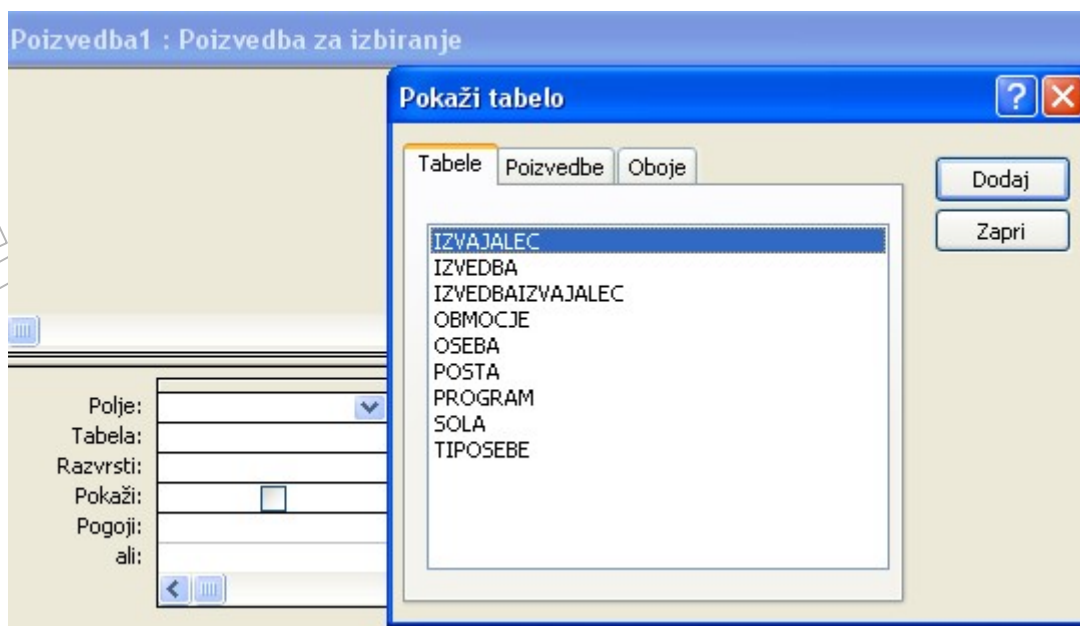
WHERE omejitve in pogoji, ki jih naštevamo z veznikom AND ali OR;

Rezultat stavka SELECT lahko shranimo tudi v tabelo, uporabimo kot del drugega stavka SELECT, kot del poročila in podobno. Najbolj pogosto pa ga uporabljamo samostojno (kot je naveden zgoraj). V tem primeru se dobljeno v obliki tabele izpiše na zaslon.

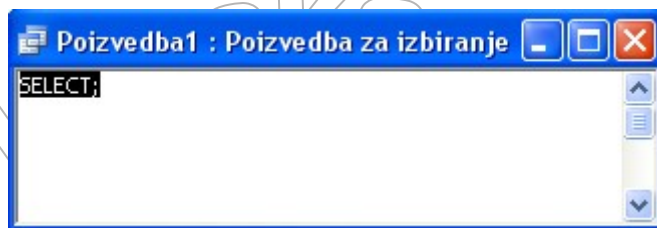
V prvem delu najbolj pogosto navedemo le imena stolpcev, ki jih želimo dobiti iz tabel. Videli pa bomo, da jih lahko kombiniramo tudi v različne izraze.



Slika 60 Izbira izdelave poizvedbe v pogledu načrta

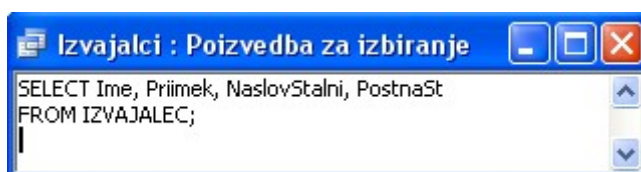


Slika 61 Izbira tabel pri izdelavi poizvedb v načinu QBE



Slika 62 Accessov urejevalnik za ukaze v SQL

Oglejmo si zgled enostavne poizvedbe. Izpišimo podatke o izvajalcih, potrebne za izdelavo naslovnih nalepk ob pošiljanju dopisov. Potrebujemo torej ime, priimek ter naslov. Podatke najdemo v tabeli IZVAJALEC. Ko naštevamo polja, ki jih želimo dobiti iz tabele, moramo uporabiti ime polja in ne oznake, ki smo jo morebiti določili z lastnostjo Napis. To lastnost smo podrobneje že opisali v poglavju 5.2.1. V stavek SELECT moramo torej vpisati 'pravo' ime polja. V izpisu pa se prikaže bodisi pravo ime polja bodisi njegova oznaka (napis), če smo jo določili. Stavek SQL, potreben za izpis podatkov, vidimo na sliki 63.



Slika 63 SQL stavek za izpis izvajalcev

Ker nismo imeli nobenih pogojev, tretjega dela (WHERE) v stavku SQL ni. Ukaz izvedemo tako, da kliknemo na rdeč klicaj v orodni vrstici ali pa Poizvedba > Zaženi. Dobimo naslednji rezultat:

	Ime	Priimek	Stalni naslov	Poštna številka
▶	EA	MIKUŽ	Pod Hrasti 3	3254
	PETRA	ČUK	Clevelandska 5	1000
	HELENA	KEBER	Samova ul. 35	2000
	MITJA	LAH	Obala 19	6000
	ŽAN	JESIH	Dunajska c. 34	1000
	JANA	SAVIČ	Slovenska c. 89	1000
*				

Slika 64 Rezultat poizvedbe

Opazimo, da nam v tabeli (slika 64) manjka še kraj (navedena je le poštna številka). Tega z enostavno poizvedbo ne moremo dobiti, saj se kraji nahajajo v drugi tabeli (v tabeli POSTA). Zato bomo polne naslove izpisali s sestavljeno poizvedbo, ki jo bomo predstavili v naslednjem poglavju.

### 5.3.2 Sestavljene poizvedbe

V sestavljenih poizvedbah poizvedujemo po podatkih iz več tabel. Pri tem moramo poznati strukturo tabel in relacije med tabelami. Pomagamo si lahko z modelom ER.

Dopolnimo primer poizvedbe iz prejšnjega poglavja tako, da bomo poleg poštne številke izpisali še kraj. V prvi del poizvedbe bomo dodali še kraj iz tabele POSTA, v drugem delu pa bomo navedli še tabelo POSTA. Ustrezni ukaz naj bi bil torej:

```
SELECT Ime, Priimek, NaslovStalni, PostnaSt, Kraj
FROM IZVAJALEC, POSTA
```

Ker pa se polje PostnaSt nahaja v obeh tabelah, je potrebno natančneje opredeliti, na katero polje mislimo. Zato moramo napisati:

```
SELECT Ime, Priimek, NaslovStalni, IZVAJALEC.PostnaSt, Kraj
FROM IZVAJALEC, POSTA
```

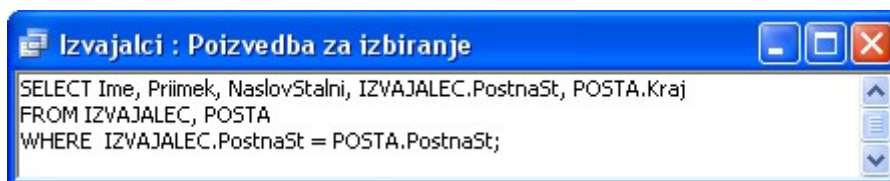
Seveda pa lahko tudi pri poljih za katera se ve, iz katerih tabel so (ker imajo enolična imena) navedemo, iz katere tabele prihajajo:

```
SELECT IZVAJALEC.Ime, IZVAJALEC.Priimek, IZVAJALEC.NaslovStalni,
IZVAJALEC.PostnaSt, POSTA.Kraj
FROM IZVAJALEC, POSTA
```

Ne glede na uporabljeno obliko nas bo izpis verjetno na prvi pogled presenetil. Izpiše se namreč zelo veliko zapisov. Če ukaz dopolnimo v:

```
SELECT IZVAJALEC.Ime, IZVAJALEC.Priimek, IZVAJALEC.NaslovStalni,
IZVAJALEC.PostnaSt, POSTA.PostnaSt, POSTA.Kraj
FROM IZVAJALEC, POSTA
```

(dodali smo, naj se izpišeta tako polje PostnaSt iz tabele IZVAJALEC, kot polje PostnaSt iz tabele POSTA) vidimo, da so se izpisale vse možne kombinacije zapisov iz obeh tabel. V poizvedbo moramo namreč dodati še pogoj, s katerim bomo dobili le tisti kraj, ki pripada poštni številki določene osebe. Za ključno besedo WHERE navedemo pogoj, naj se izpišejo tisti zapisi, pri katerih se poštna številka v tabeli IZVAJALEC ujema s poštno številko v tabeli POSTA. Dopolnjeno poizvedbo (kjer smo polje PostnaSt morali opremiti s tabelo iz katere izvira, za polje Kraj pa to ne bi bilo potrebno) vidimo na sliki 65, njen rezultat pa na sliki 66.



Slika 65 Sestavljena poizvedba

	Ime	Priimek	Stalni naslov	Poštna	Kraj
	LEA	MIKUŽ	Pod Hrasti 3	3254	Podčetrtek
	PETRA	ČUK	Clevelandska 5	1000	Ljubljana
	HELENA	KEBER	Samova ul. 35	2000	Maribor
	MITJA	LAH	Obala 19	6000	Koper/Capodistria
	ŽAN	JESIH	Dunajska c. 34	1000	Ljubljana
▶	JANA	SAVIČ	Slovenska c. 89	1000	Ljubljana

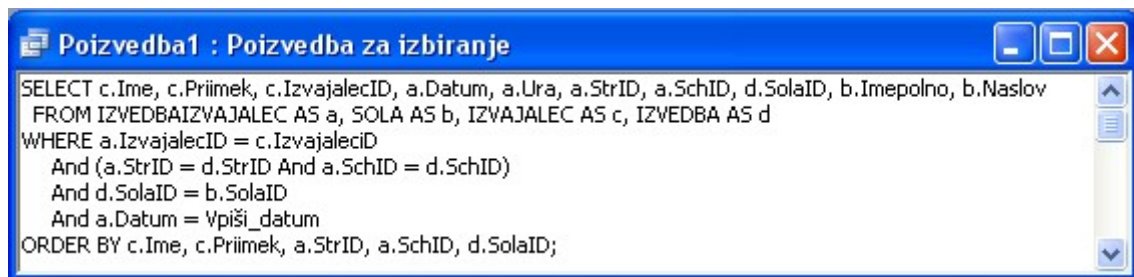
Zapis: 6 od 6

Slika 66 Rezultat sestavljene poizvedbe

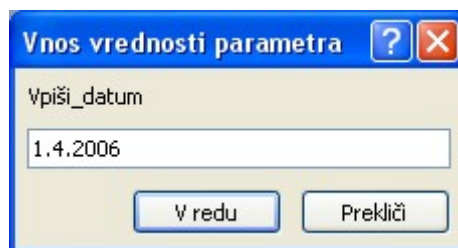
Za poizvedbe v nadaljevanju bomo uporabljali tudi poizvedbe s parametrom. Poizvedba s parametrom je poizvedba, ki ob izvedbi prikaže pogovorno okno, ki nas sprašuje po določenih pogojih za pridobivanje zapisov. Naredimo jo tako, da pod WHERE napišemo npr. pogoj `a.Datum = Vpiši_datum`. Pri tem `a` označuje neko tabelo, ki smo jo določili pod FROM. Datum je atribut (ime stolpca) v tej tabeli `a`. `Vpiši_datum` je ime parametra, katerega vrednost bo poizvedba zahtevala ob zagonu. Želimo torej izpisati podatke, ki imajo ta atribut enak datumu, ki ga bo uporabnik vpisal v pogovorno okno. Primer pogovornega okna vidimo na sliki 68.

Preglejmo vprašanja iz poglavja 5.1.1. in si nanje odgovorimo s pomočjo poizvedb.

Prvo vprašanje se glasi: »**Kateri izvajalec ima danes izvedbo in na kateri šoli?**«. Želimo torej izpis izvajalcev, ki imajo na določen datum izvedbo in zraven še izpis šole, na kateri je ta izvedba. Glavni pogoj je torej datum izvedbe, ki se nahaja v tabeli IZVEDBAIZVAJALEC. V njej je tudi podatek, kateri izvajalec vodi izvedbo in na kateri šoli je to. Podatki bodo torej izvirali iz tabel IZVEDBAIZVAJALEC, IZVAJALEC in SOLA. Poizvedbo vidimo na sliki 67, njen rezultat pa na sliki 69. Poizvedbe ne bomo podrobneje razčlenjevali, ker se od bralca pričakuje vsaj osnovno poznavanje jezika SQL. Če temu ni tako, si lahko osnove prebere v [4], zahtevnejšim bralcem pa priporočamo [5]. V nadaljevanju bomo navedli le vprašanje, osnovna pojasnila, ukaz v SQL in rezultat poizvedbe, ki je odgovor na vprašanje.



Slika 67 Poizvedba za odgovor na prvo vprašanje



Slika 68 Okno za vnos datuma



	Ime	Priimek	IzvajalecID	Datum	Ura	StrID	SchID	ID šole	
▶	EA	MIKUŽ	1	1.4.2006	14:00	03	015	103	E
	PETRA	ČUK	2	1.4.2006	14:00	04	030	103	E
	ŽAN	JESIH	5	1.4.2006	8:00	04	006	102	B

Slika 69 Rezultat poizvedbe

2. Katere izvedbe ima izvajalec v nekem časovnem obdobju?

Tokrat bo parametrov, ki jih bo uporabnik vnesel ob izvedbi poizvedbe, več. To bodo ime in priimek izvajalca, ter datum začetka in konca obdobja. Izpisali bomo glavne podatke o posamezni izvedbi kot so StrID, SchID, IzvajalecID, IzvedbaID, Datum, Ura ter SolaID. Če bo uporabnik potreboval še ime šole in njen naslov, ju bomo z dopolnitvijo poizvedbe preko polja SolaID pridobili iz tabele SOLA.

Rezultatov nismo uredili (kar bi lahko kot prej naredili že s samo poizvedbo z določilom ORDER BY). V Accessu namreč uporabnik lahko rezultat poizvedbe enostavno uredi po kateremkoli stolpcu. Uporabi desni klik na ime stolpca in izbere naraščajočo ali padajočo razvrstitev. Rezultat, prikazan na sliki 71, se nanaša na parametre LEA, MIKUŽ in obdobje 10.3.2006 – 15.5.2006.

```

SELECT a.StrID, a.SchID, a.IzvajalecID, a.IzvedbaID, a.Datum,a.Ura, b.SolaID
FROM IZVEDBAIZVAJALEC AS a, IZVEDBA AS b, IZVAJALEC AS c
WHERE a.IzvajalecID = c.IzvajalecID
  And c.Ime = Ime_izvajalca
  And c.Priimek = Priimek_Izvajalca
  And a.Datum Between Začetni_datum And Končni_datum
  And a.StrID=b.StrID And a.SchID=b.SchID;
    
```

Slika 70 Poizvedba za odgovor na drugo vprašanje

	StrID	SchID	IzvajalecID	IzvedbaID	Datum	Ura	ID šole
	01	007	1	1	21.3.2006	10:00	101
	01	007	1	2	30.3.2006	9:00	101
▶	03	015	1	1	1.4.2006	14:00	103

Zapis: 3 od 3

Slika 71 Rezultat poizvedbe

### 3. Katere šole so v določeni območni enoti?

Vhodni parameter bo območna enota. Izpisali bomo naslednje podatke o šoli: SolaID, ImePolno, Naslov in Enota. Rezultat na sliki 73 se nanaša na parameter 1.

```

SELECT a.SolaID, a.ImePolno, a.Naslov, b.Enota
FROM SOLA AS a, OBMOCJE AS b
WHERE a.ObmocjeID = b.ObmocjeID
  AND a.ObmocjeID = ID_območja;
    
```

Slika 72 Poizvedba za odgovor na tretje vprašanje



SolalID	ImePolno	Naslov	Enota
101	Biotehniški izobraževalni center Ljubljana	Cesta v Mestni log 47	Ljubljana
102	Biotehniški izobraževalni center Ljubljana	Ižanska c. 10	Ljubljana
103	Ekonomška šola Ljubljana	Prešernova 6	Ljubljana
104	Gimnazija Bežigrad	Peričeva 4	Ljubljana
105	Gimnazija EURO šola Ljubljana	Litistrojska c. 40	Ljubljana
106	Gimnazija in ekonomska srednja šola Trb	Gimnazijska 10	Ljubljana
107	Gimnazija Jožeta Plečnika Ljubljana	Šubičeva 1	Ljubljana

Slika 73 Rezultat poizvedbe

#### 4. Kateri izvajalci na določen datum nimajo načrtovane nobene izvedbe?

Vhodni parameter bo datum. Izpisali bomo podatke o izvajalcu in sicer njegov ID, ime, priimek ter telefonsko številko. Rezultat na sliki 75 se nanaša na parameter 23. 3. 2006.

```
SELECT a.IzvajalecID, a.Ime, a.Priimek, Tel1
FROM IZVAJALEC AS a
WHERE NOT EXISTS (SELECT *
FROM IZVEDBAIZVAJALEC AS b
WHERE b.IzvajalecID=a.IzvajalecID And b.Datum = Vnesi_datum);
```

Slika 74 Poizvedba za odgovor na četrto vprašanje

IzvajalecID	Ime	Priimek	Mobi
1	LEA	MIKUŽ	031/578-356
2	PETRA	ČUK	041/648-478
3	HELENA	KEBER	031/078-966
4	MITJA	LAH	041/945-675
5	ŽAN	JESIH	051/152-468

Slika 75 Rezultat poizvedbe

#### 5. Ali ima določen izvajalec danes izvedbo in katero?

Vhodni parameter bo ime in priimek izvajalca. Ker bomo uporabniku dovolili, da bo lahko vnesel tudi poljuben datum, bomo uporabnost te poizvedbe še razširili. Poizvedba bo torej povedala, ali ima določen izvajalec na navedeni dan izvedbo in katero. Izpisali bomo podatke o izvedbi, ki izvedbo enolično definirajo in sicer StrID, SchID, IzvedbaID, Ura, SolaID in ImePolno. Rezultat na sliki 77 se nanaša na parametre ŽAN JESIH 1. 4. 2006.

```

Izvedba_izvajalca_na_datum_vpr5 : Poizvedba za izbiranje
SELECT a.StrID, a.SchID, a.IzvedbaID, a.Ura, d.SolaID, d.ImePolno
FROM IZVEDBAIZVAJALEC AS a, IZVAJALEC AS b, IZVEDBA AS c, SOLA AS d
WHERE a.IzvajalecID = b.IzvajalecID
  And (a.StrID = c.StrID And a.SchID=c.SchID)
  And c.SolaID = d.SolaID And a.Datum= Vnesi_datum
  And b.Ime = Ime_izvajalca
  And b.Priimek = Priimek_izvajalca;

```

Slika 76 Poizvedba za odgovor na peto vprašanje

StrID	SchID	IzvedbaID	Ura	SolaID	ImePolno
104	006	1	8:00	102	Biotehniški izobraževalni center Ljub

Zapis: 1 od 1

Slika 77 Rezultat poizvedbe

## 6. Kdo je koordinator na določeni šoli?

Zaradi enostavnosti bo vhodni parameter kar ID šole. Bolj smiselno bi bilo, da bi uporabnik vpisal ime šole, oziroma da bi mu omogočili izbiro imena šole iz seznama. Vendar slednje lahko prepustimo uporabniškemu vmesniku, od koder bi potem zlahka pridobili omenjeni ID šole. Izpisali bomo ID osebe, njeno ime in priimek ter njen TipID. Omenimo, da nam gre bolj za ilustracijo, kako je možno z poizvedbami pridobiti odgovore na vsa, v začetni analizi, navedena vprašanja uporabnikov, ne pa toliko za uporabnost poizvedb. "Prava" aplikacija bi uporabniku prikazala nazive tipov oseb in ne šifer. A tovrstne izboljšave je dokaj enostavno narediti preko kombinacije z drugimi poizvedbami in z uporabniškim vmesnikom.

Glede na to, da želimo izpisati koordinatorja, moramo v poizvedbi postaviti omejitve za TipID. Iščemo zapise, ki imajo vrednost polja TipID enako 2 ali 3. To pomeni, da je oseba koordinator ali pa je ravnatelj, ki je hkrati tudi koordinator. Rezultat na sliki 79 se navezuje na šolo z identifikacijsko številko 102. Vidimo, da sta na tej šoli dve koordinatorici. Prva je Maja Hrust, ki je samo koordinatorica, druga pa Lidija Kos, ki je tudi ravnateljica te šole.

```

Koor dinatorji_določene_šole : Poizvedba za izbiranje
SELECT OsebaID, Ime, Priimek, TipID
FROM OSEBA
WHERE SolaID=Vpiši_ID_šole
  And (TipID="2" Or TipID="3");

```

Slika 78 Poizvedba za odgovor na šesto vprašanje

OsebaID	Ime	Priimek	TipID
1	MAJA	HRUST	2
6	LIDIJA	KOS	3

Zapis: 2 od 2

Slika 79 Rezultat poizvedbe

## 7. Na kateri šoli je določena oseba koordinator?

Vhodna parametra bosta ime in priimek koordinatorja. Izpisali bomo podatke o šoli s katere prihaja ta koordinater in sicer ID šole, njeno polno ime in naslov. Rezultat poizvedbe (slika 80) vidimo na sliki 81.

```

Šola_določenega_koordinatorja_vpr7 : Poizvedba za izbiranje
SELECT b.SolaID, ImePolno, Naslov
FROM SOLA AS a, OSEBA AS b
WHERE a.SolaID = b.SolaID
And b.Ime = Ime_koordinatorja
And b.Priimek = Priimek_koordinatorja;
    
```

Slika 80 Poizvedba za odgovor na sedmo vprašanje

SolaID	ImePolno	Naslov
107	Gimnazija Jožeta Plečnika Ljubljana	Šubičeva 1

Zapis: 1 od 1

Slika 81 Rezultat poizvedbe

### 8. Koliko izvedb ima določen koordinater?

Vhodna parametra bosta spet ime in priimek koordinatorja. Ker je praviloma število izvedb dokaj majhno, bomo raje izpisali osnovne podatke o izvedbah. Uporabnik bo število izvedb zlahka preštel sam.

```

Izvedbe_določenega_koordinatorja_vpr8 : Poizvedba za izbiranje
SELECT StrID, SchID, a.SolaID, ImePolno, Naslov
FROM IZVEDBA as a, SOLA as b, OSEBA as c
WHERE a.SolaID = b.SolaID
AND a.OsebaID = c.OsebaID
AND c.Ime = Ime_koordinatorja
AND c.Priimek = Priimek_koordinatorja;
    
```

Slika 82 Poizvedba za odgovor na osmo vprašanje

StrID	SchID	ID šole	ImePolno	Naslov
01	011	104	Gimnazija Bežigrad	Peričeva 4
02	026	104	Gimnazija Bežigrad	Peričeva 4

Zapis: 2 od 2

Slika 83 Rezultat poizvedbe

### 9. Kateri izvajalec izvaja določeno izvedbo?

Vhodna parametra bosta StrID in SchID. S tem bomo dobili informacijo tudi o morebitnih ponovitvah te izvedbe. Zato parametra IzvedbaID ne bomo vnašali. Izpisali bomo ID izvajalca, ID izvedbe, njegovo ime in priimek ter telefonsko številko. Rezultat na sliki 85 ustreza parametroma 02 in 023.

```

Izvajalci_določene_izvedbe_vpr9 : Poizvedba za izbiranje
SELECT a.IzvajalecID,b.IzvedbaID, a.Ime, a.Priimek, a.Tel1
FROM IZVAJALEC AS a, IZVEDBAIZVAJALEC AS b
WHERE a.IzvajalecID = b.IzvajalecID
      And b.StrID = Vnesi_StrID
      And b.SchID = Vnesi_SchID;

```

Slika 84 Poizvedba za odgovor na deveto vprašanje

	IzvajalecID	IzvedbaID	Ime	Priimek	Mobi
▶	4	1	MITJA	LAH	041/945-675

Zapis: 1 od 1

Slika 85 Rezultat poizvedbe

### 10. Na kateri šoli potekajo izvedbe vzporedno?

V tej poizvedbi ne bomo imeli vhodnega parametra. Izpisati želimo tiste šole, ki imajo v tabeli IZVEDBAIZVAJALEC več kot en zapis z istim datumom in uro izvedbe. Takrat se na isti šoli vzporedno izvajata dve vzorčni enoti. Običajno to pomeni, da se na šoli izvaja raziskava za dva učna programa hkrati. Druga možnost bi bila ta, da je v vzorčni enoti preveč učencev, da bi se med raziskavo vsi nahajali v enem prostoru. To je sicer malo verjetno saj mednarodni vzorec ne presega števila 35 učencev. A pri dodatnih nacionalnih raziskavah se to lahko zgodi, saj tu velikost vzorca (število učencev) lahko doseže tudi vrednost 100.

```

Šole_z_vzporednimi_izvedbami : Poizvedba za izbiranje
SELECT i1.SolaID, i2.Datum, i2.Ura, s.ImePolno
FROM IZVEDBA AS i1, IZVEDBAIZVAJALEC AS i2, SOLA AS s
WHERE i1.StrID = i2.StrID
      AND i1.SchID = i2.SchID
      AND i1.SolaID = s.SolaID
GROUP BY i1.SolaID,
          i2.Datum,
          i2.Ura,
          s.ImePolno
HAVING count(*) > 1
ORDER BY s.ImePolno;

```

Slika 86 Poizvedba za vzporedne izvedbe

Rezultat na sliki 87 nam pove, da obstajata dve šoli, kjer sta sočasno potekali vsaj dve izvedbi.

	ID šole	Datum	Ura	ImePolno
	103	1.4.2006	14:00	Ekonomska šola Ljubljana
▶	104	14.9.2006	8:00	Gimnazija Bežigrad

Zapis: 2 od 2

Slika 87 Rezultat poizvedbe

V odgovoru na sliki 87 pogrešamo informacijo, katere vzorčne enote se izvajajo ob izpisanem terminu. Naredimo zato še eno poizvedbo, kjer si izpišimo vse izvedbe na določen datum. Izvedbe razvrstimo po vrednosti polj SolaID in Ura, da bomo lahko hitro našli vzporedne izvedbe. Rezultat na sliki 89 pripada parametru 1. 4. 2006.

```
SELECT b.StrID, b.SchID, SolaID, a.Datum, a.Ura
FROM IZVEDBAIZVAJALEC AS a, IZVEDBA AS b
WHERE a.StrID = b.StrID And a.SchID = b.SchID And Datum = vpisi_datum
ORDER BY SolaID, Datum, Ura;
```

Slika 88 Poizvedba za izvedbe na določen datum

StrID	SchID	ID šole	Datum	Ura
04	006	102	1.4.2006	8:00
04	030	103	1.4.2006	14:00
03	015	103	1.4.2006	14:00

Slika 89 Rezultat poizvedbe

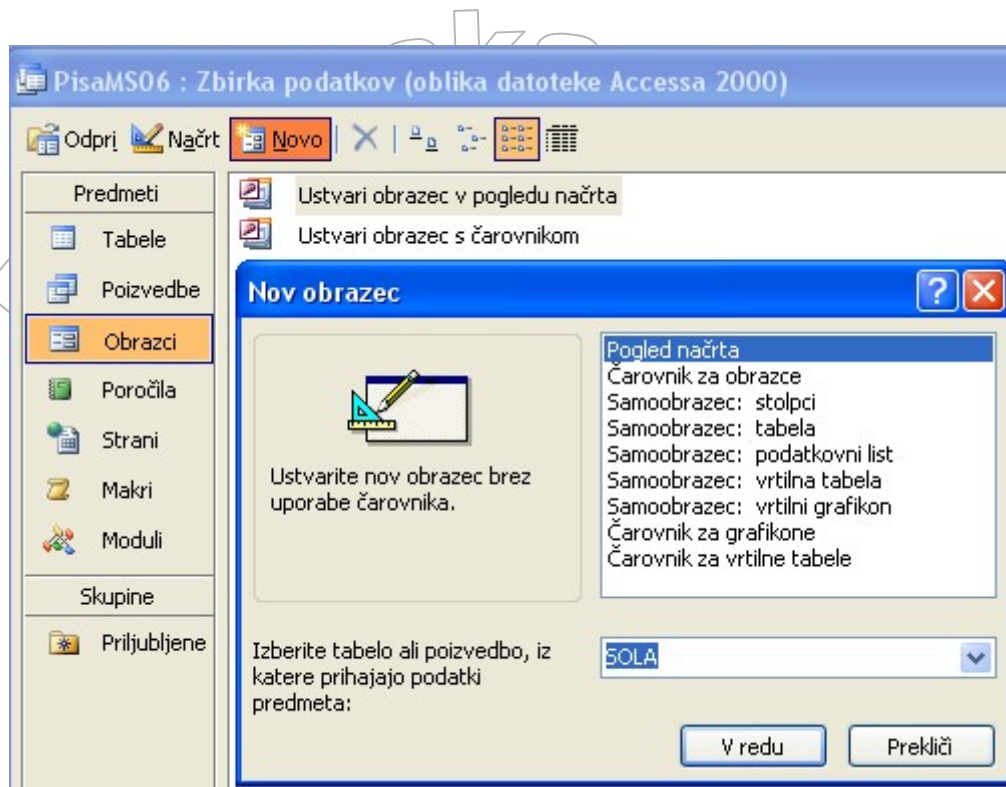
## 5.4 Izdelava obrazcev

V poglavju 5.2.5. smo podatke v tabele vnesli neposredno v tabele. Program Access nam omogoča tudi preprostejši vnos preko obrazcev oz. zaslonskih mask. Preko obrazca lahko vnašamo podatke v posamezno tabelo ali pa v več tabel hkrati. Obrazec je v bistvu grafični vmesnik med uporabnikom in tabelami v bazi podatkov. Preko njega podatke vnašamo, brišemo, popravljamo itd. Nanje je možno dodajati tudi različne kontrole in gumbe z dodatnimi funkcijami, kot na primer za tiskanje podatkov.

Obrazec lahko izdelamo v pogledu načrta ali pa s pomočjo čarovnika. Obrazce razdelimo na preproste in kompleksne. Prvi so preprosti, ker preko njih vnašamo podatke le v eno tabelo, kompleksni pa so tisti, preko katerih vnašamo podatke v več tabel. Preprost obrazec bomo izdelali v pogledu načrta, kompleksnega pa s pomočjo čarovnika.

### 5.4.1 Preprosti obrazci

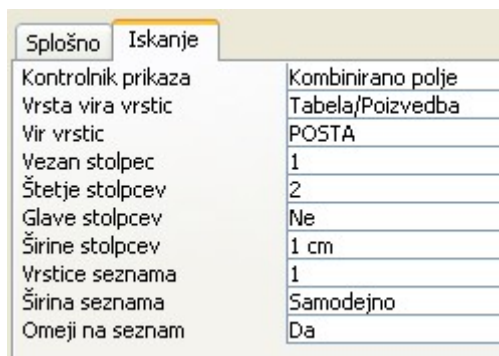
S pomočjo preprostih obrazcev vnašamo podatke v posamezno tabelo. Naredimo obrazec za vnos podatkov v tabelo SOLA. V osnovnem oknu na levi izberemo predmet Obrazci. V menijski vrstici kliknemo na ukaz Novo. V oknu, ki se pojavi, izberemo Pogled načrta in spodaj tabelo SOLA (slika 90).



Slika 90 Izberemo Pogled načrta za izdelavo obrazca za tabelo SOLA

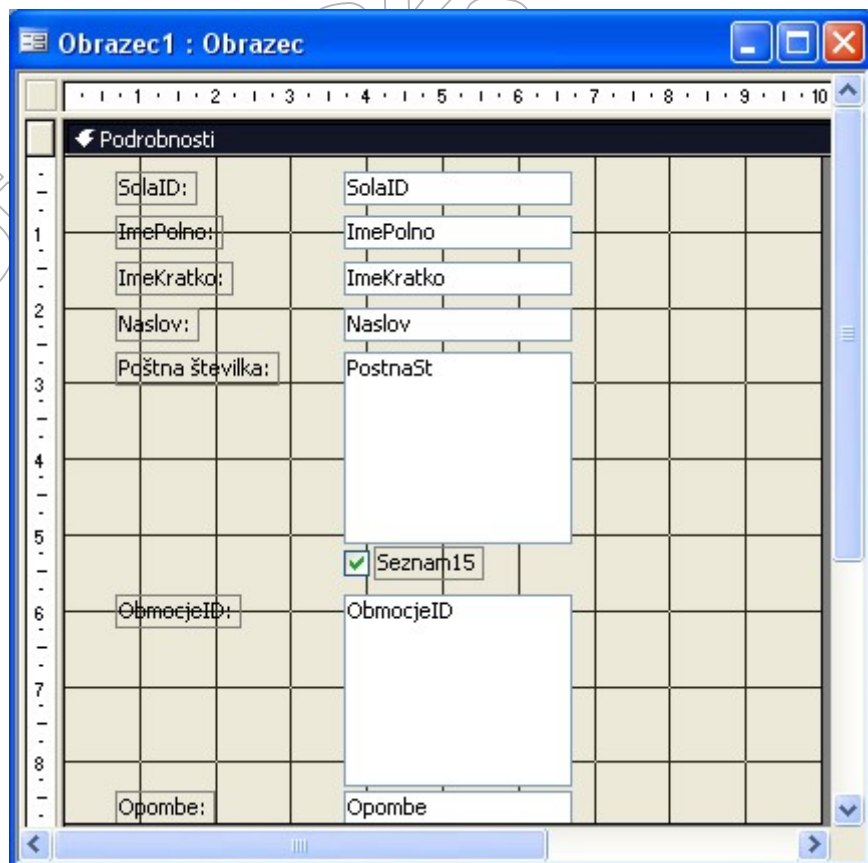
Program nam da na izbiro polja, ki jih želimo imeti v obrazcu. Izberemo vsa, saj želimo s tem obrazcem vnesti vse podatke o šoli. Po potrditvi izbire se odpre okno, ki je prikazano na sliki 92. Če nam položaj posameznega polja ne ustreza, ga lahko premaknemo drugam. Prav tako lahko spreminjamo napise poleg polj.

Za posamezno polje lahko poskrbimo, da njegovo vrednost določamo s pomočjo izbirnega seznama. Tako smo storili s polji PostnaSt in ObmocjeID. Slika 91 prikazuje lastnosti, ki smo jih določili polju PostnaSt. Z njimi smo v obrazcu dobili izbirni seznam, v katerem si lahko izberemo poštno številke. Ta seznam se napolni avtomatično iz tabele POSTA.



Slika 91 Lastnosti polja PostnaSt v tabeli SOLA

Med obliko načrta (slika 92) in pogledom obrazca (slika 93) preklapljamo z ikonama  in , ki se nahajata v orodni vrstici programa Access.



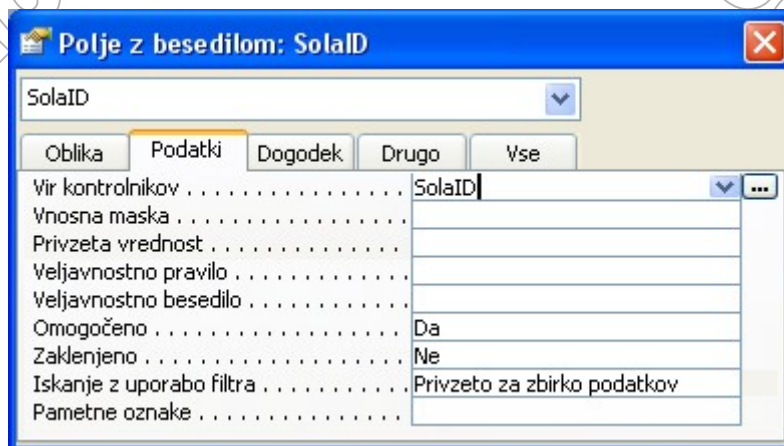
Slika 92 Okno za urejanje obrazca

Zgornji načrt smo nekoliko uredili in preoblikovali in izdelali preprost obrazec:

Slika 93 Preprost obrazec za vnos podatkov v tabelo SOLA

Tudi pri izdelavi obrazca imamo možnost številnih nastavitev. Do njih pridemo tako, da v pogledu načrta obrazca (slika 92) kliknemo z desno tipko miške in iz menija izberemo ukaz

Lastnosti. Prikaže se okno, kot ga vidimo na sliki 94. Na vrhu izbiramo posamezne elemente, ki se nahajajo na obrazcu, spodaj pa imamo možnost nastavitve lastnosti zgoraj izbranega elementa. V diplomski nalogi se ne bomo spuščali v podrobnosti vseh teh lastnosti, temveč bomo sproti opisali le tiste, ki jih bomo uporabili v naših obrazcih.



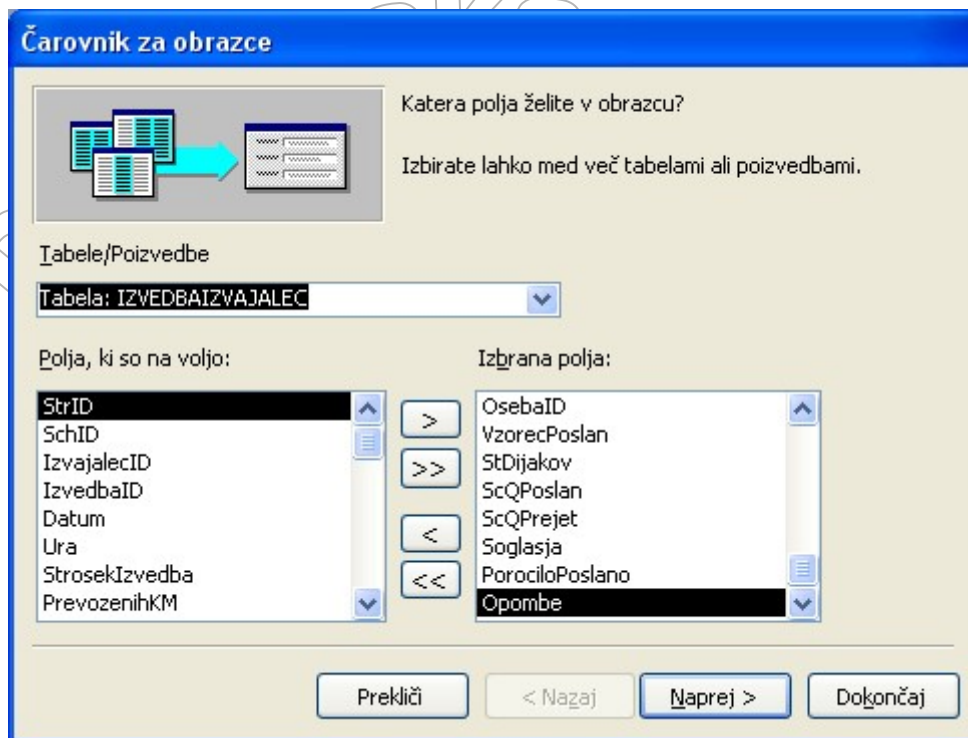
Slika 94 Okno z lastnostmi za posamezno polje v obrazcu

#### 5.4.2 Kompleksni obrazci

Pri vnašanju podatkov v povezane tabele, ki smo jih opisali v poglavju 5.2.5., je potrebno paziti, saj nam program ne dovoli vnosa povezanih podatkov, ki še ne obstajajo v primarni tabeli. Odprtih je bilo potrebno imeti več tabel ali pa smo si morali zapomniti vsebino primarnih tabel. Vsekakor tak način vnosa uporabnikom ne ustreza. Želeli bi imeti vmesnik, kamor bi lahko vnašali podatke v več tabel hkrati. Poleg tega želimo imeti lahko dostopen podatek o tem, kateri zapisi se nahajajo v primarni tabeli. Vse to lahko dosežemo z izdelavo kompleksnega obrazca.

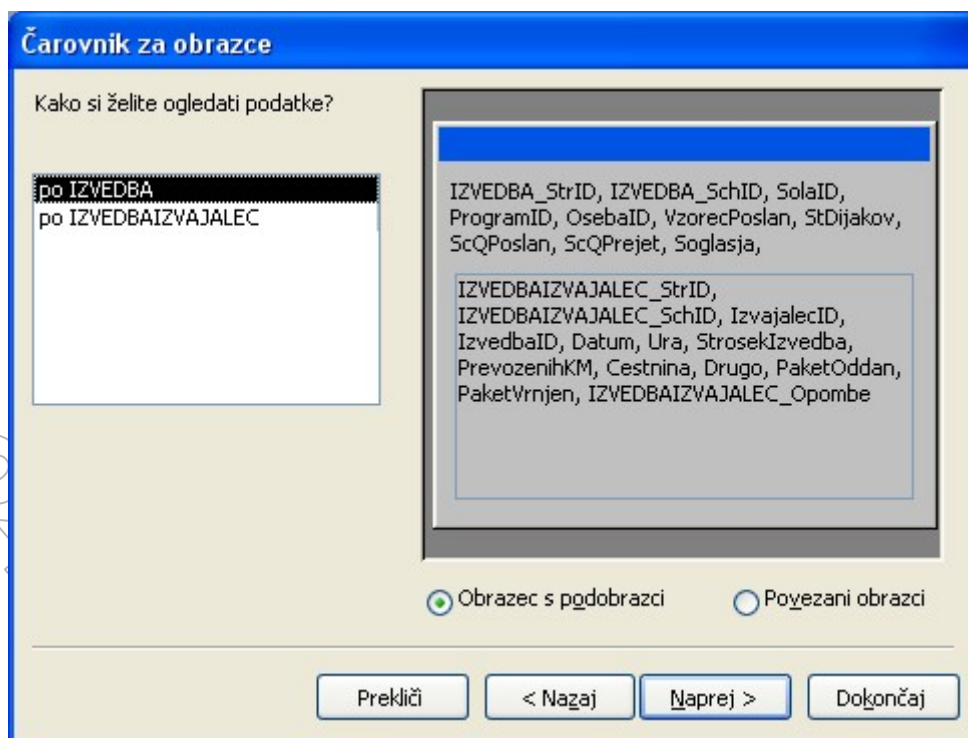
Tokrat bomo obrazec izdelali s pomočjo čarovnika. Obrazec bo namenjen vnosu izvedb v tabelo IZVEDBA s podobrazcem za vnos v tabelo IZVEDBAIZVAJALEC. Pot do izdelave obrazca je enaka kot za izdelavo v pogledu načrta, le da na sliki 90 izberemo izbiro »Čarovnik za obrazce«. V naslednjem koraku (slika 95) izberemo vsa polja iz tabele IZVEDBA in tabele IZVEDBAIZVAJALEC.





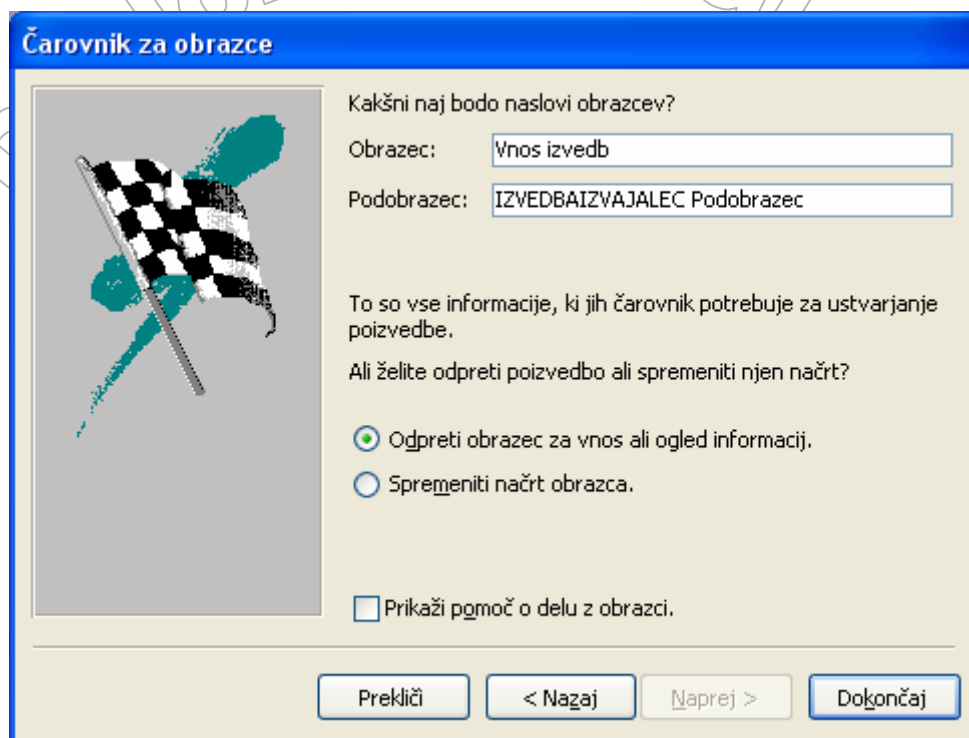
Slika 95 Izberemo polja za obrazec in podobrazec

V naslednjem koraku (slika 96) določimo katera izmed tabel je glavni obrazec, katera pa podobrazec. V nadaljevanju nas program vpraša, kakšno postavitev želimo v podobrazcu. Mi smo izbrali »Tabela«. Slika tega koraka ni prikazana.



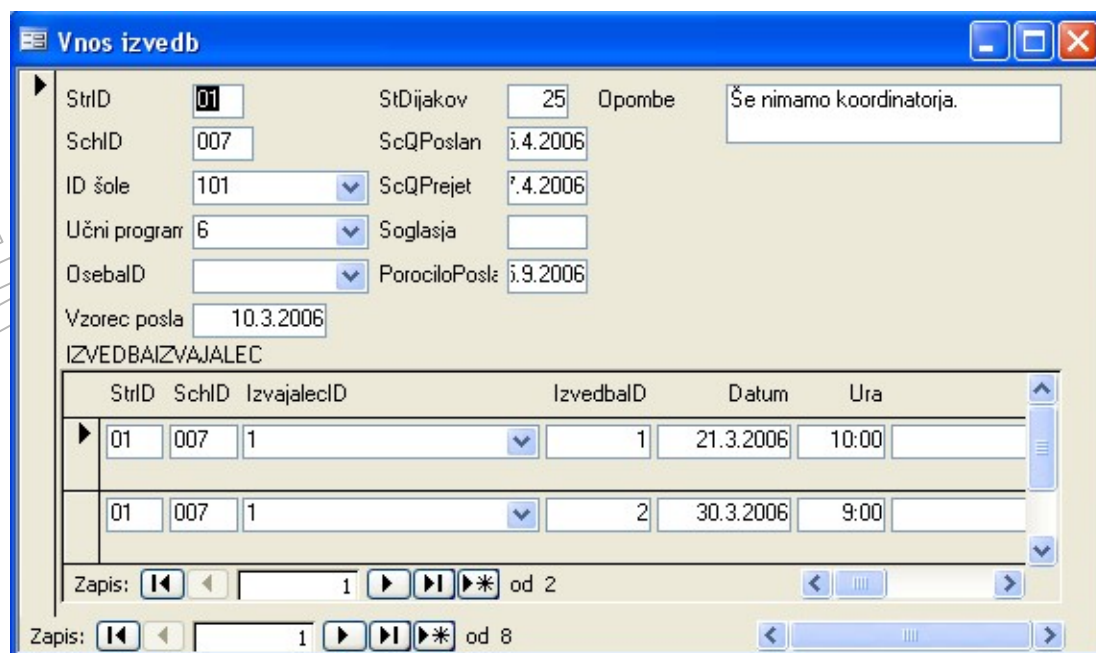
Slika 96 Določimo podobrazec IZVEDBAIZVAJALEC

Zatem določimo še slog. Izbrali smo standarden slog. Slika ni prikazana. Pokaže se zadnje okno (slika 97). Vpišemo ime obrazca in podobrazca ter izberemo »Dokončaj«.



Slika 97 Vnesemo ime obrazca

Prikaže se obrazec v končni obliki (slika 98). Obrazec lahko v pogledu načrta še ročno oblikujemo in spreminjamo ter določamo različne lastnosti.

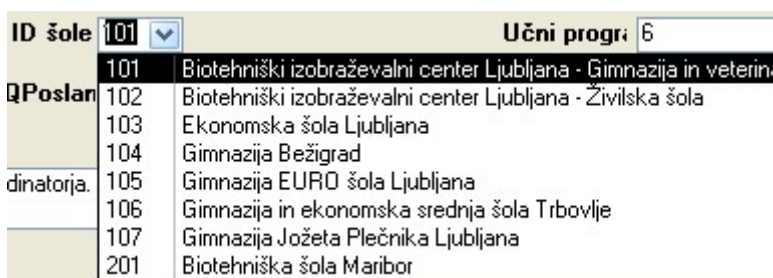


Slika 98 Obrazec v pogledu načrta

Za lažji vnos smo naredili kombinirano polje za vnos ID šole in mu določili lastnosti, ki jih vidimo spodaj. Kombinirano polje je sestavljeno iz dveh polj: IDŠole (torej podatek, ki ga vnašamo) in ime, ki pove ime šole. Na ta način bodo uporabniki zlahka vnesli pravilni IDŠole, saj bodo v izbirnem seznamu le poiskali ustrezno ime šole. Širina stolpcev pove, koliko naj bodo široki stolpci, ko pritisnemo na puščico izbirnega seznama. Vrstice seznama določajo, koliko vrstic naj bo hkrati vidnih na prikazanem seznamu. Širina seznama pa je širina celotnega seznama in mora biti enak vsoti širin stolpcev.

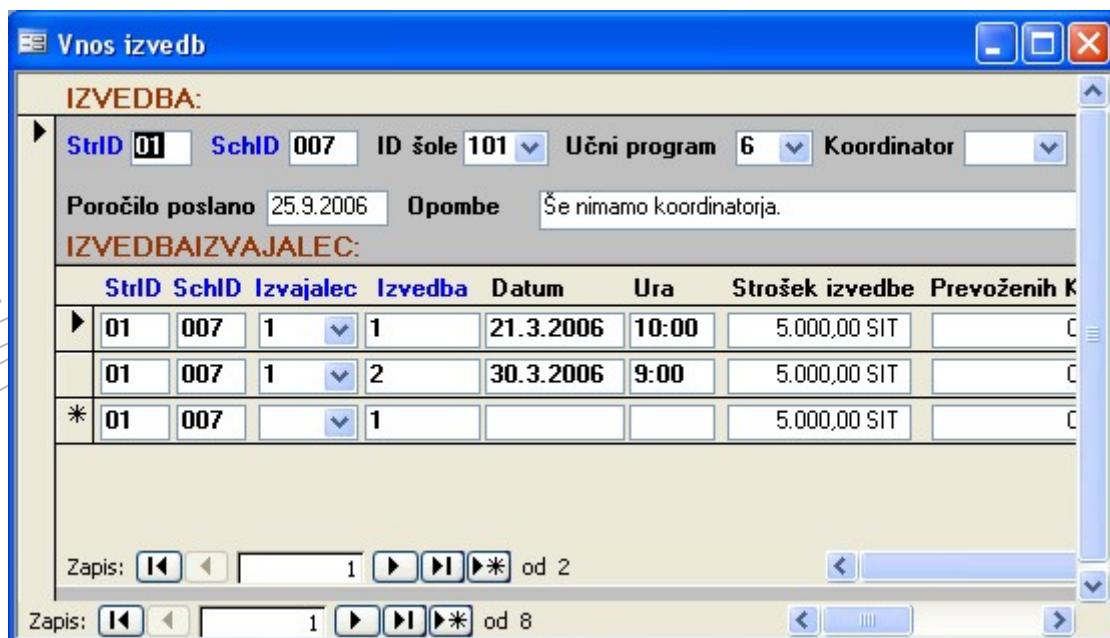
Širine stolpcev . . . . .	1 cm;8 cm
Vrstice seznama . . . . .	8
Širina seznama . . . . .	9 cm

Rezultat zgornjih nastavitvev vidimo na sliki 99. Ob kliku na puščico polja ID šole se nam prikaže seznam širine 9 cm z osmimi vrsticami.



Slika 99 Kombinirano polje za polje ID šole.

Obrazec (iz slike 98) smo razširili in si polja na glavnem obrazcu razvrstili v dve daljši vrstici. Del končnega obrazca vidimo na sliki 100.

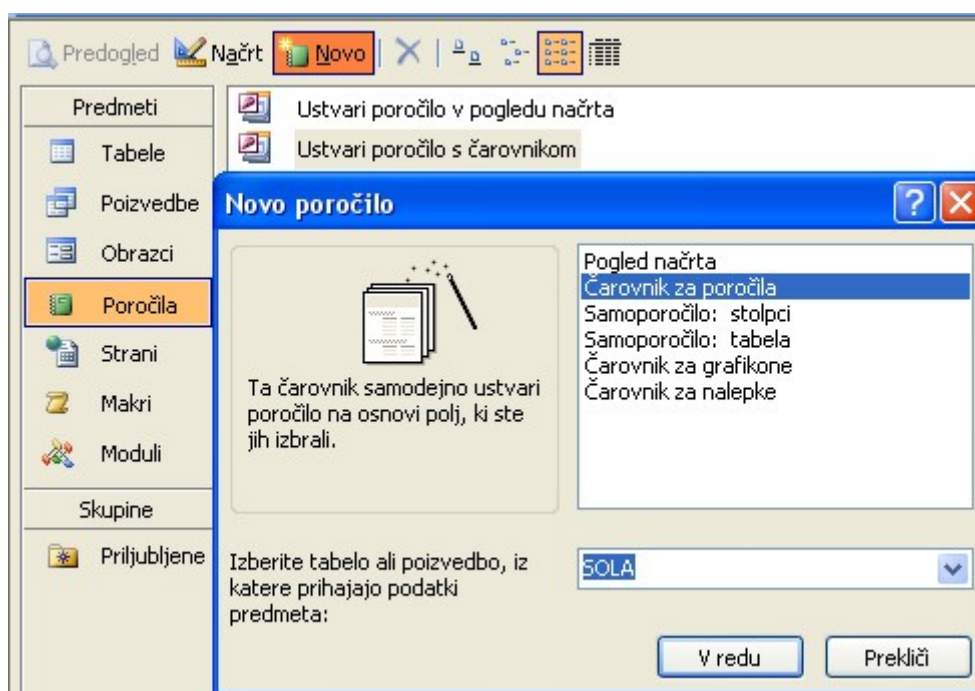


Slika 100 Del končnega obrazca s podobrzcem

## 5.5 Izdelava poročil

Poročila so namenjena prikazovanju podatkov na zaslonu oz. tiskanju podatkov iz podatkovne baze. Temeljijo na podatkih neke tabele ali poizvedbe, ponujajo pa tudi možnost dodatnega oblikovanja, dodajanja besedila, formul itd.

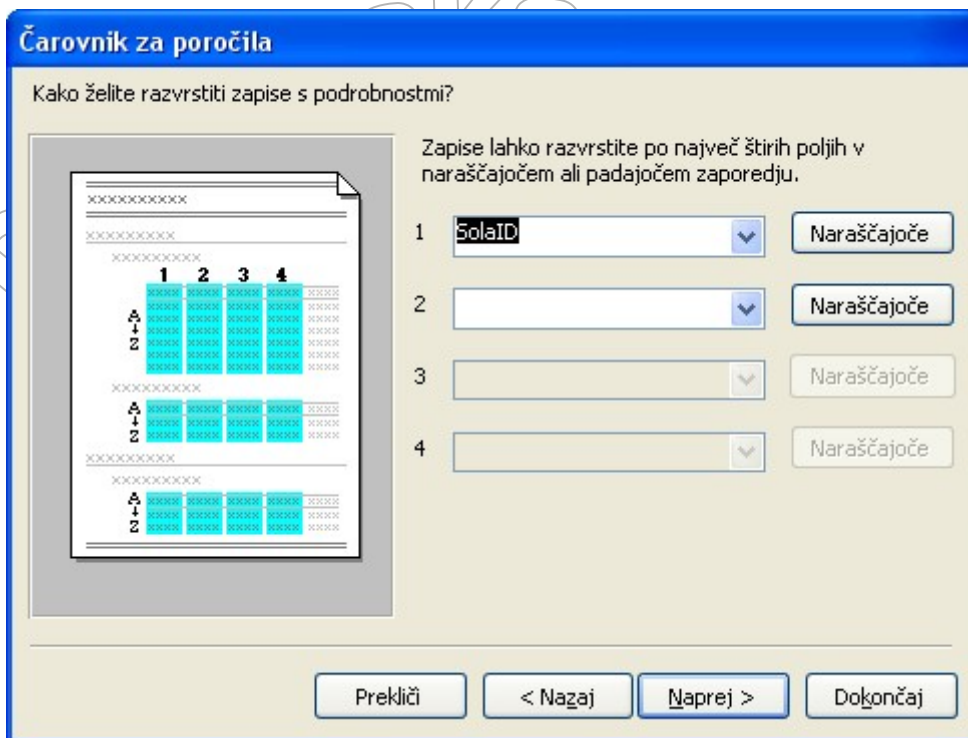
Izdelajmo poročilo s podatki o šolah s pomočjo čarovnika. V osnovnem oknu programa Access izberemo »Poročila« in zgoraj »Novo« (slika 101). V oknu, ki se odpre, izberemo »Čarovnik za poročila«. Določimo tudi vir podatkov za poročilo. V našem primeru je to tabela SOLA. V naslednjem koraku podobno kot za obrazec (slika 96) dodamo polja, ki jih želimo prikazati na poročilu.



Slika 101 Izdelava poročila s pomočjo čarovnika

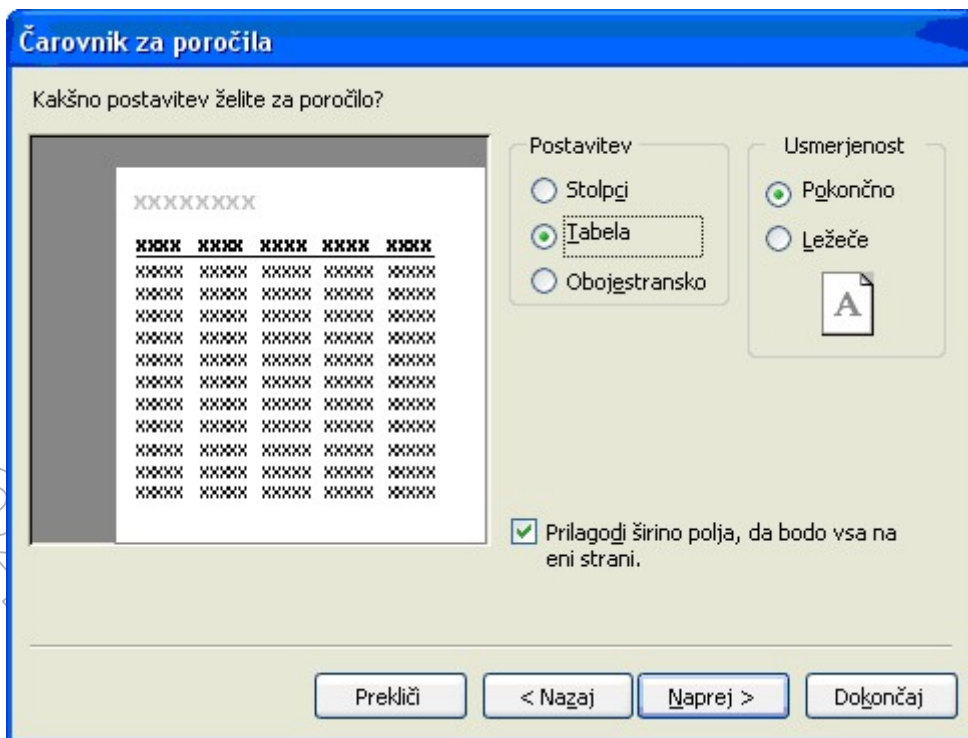
V naslednjem koraku (slika ni prikazana) je možno podatke združiti v skupine glede na vrednosti v določenem polju. V tem primeru to za nas ni smiselno, razen, če bi želeli imeti podatke o šolah združene po območjih.

V poročilu so zapisi lahko urejeni. Kot vidimo na sliki 102, je ključ pri urejanju lahko sestavljen iz največ štirih polj. Mi smo uporabili razvrstitev šol po naraščajoči vrednosti identifikacijskih števil.



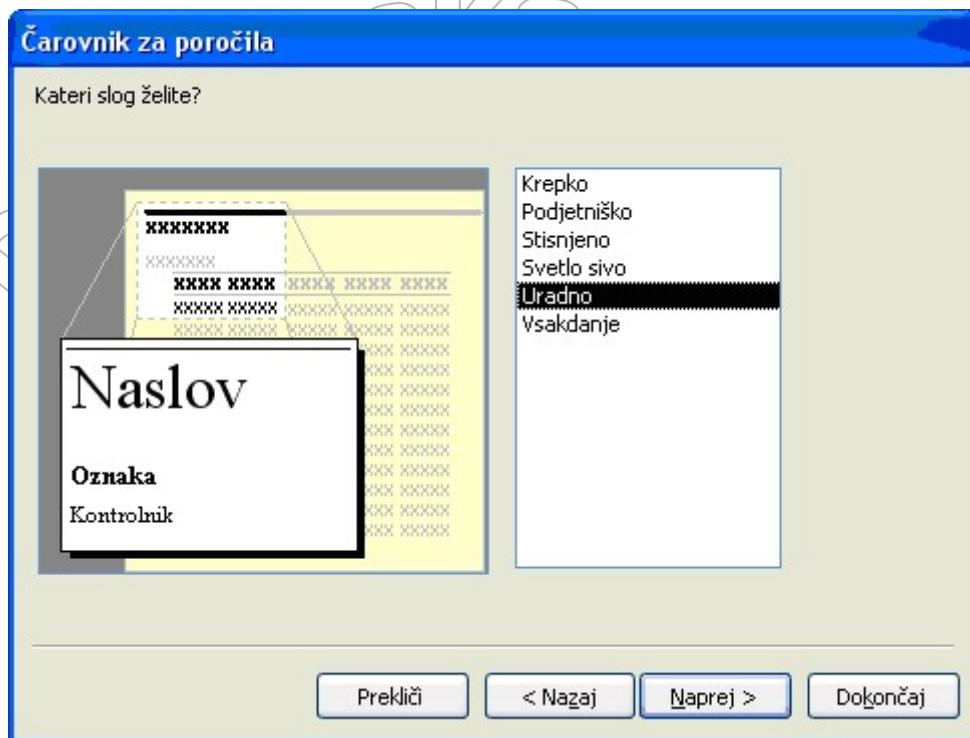
Slika 102 Šole razvrstimo naraščajoče po SolaID

Zatem čarovnik zahteva podatek, kakšno postavitev in usmerjenost poročila želimo (slika 103). Izbrali smo postavitev v obliki tabele in pokončno postavitev.



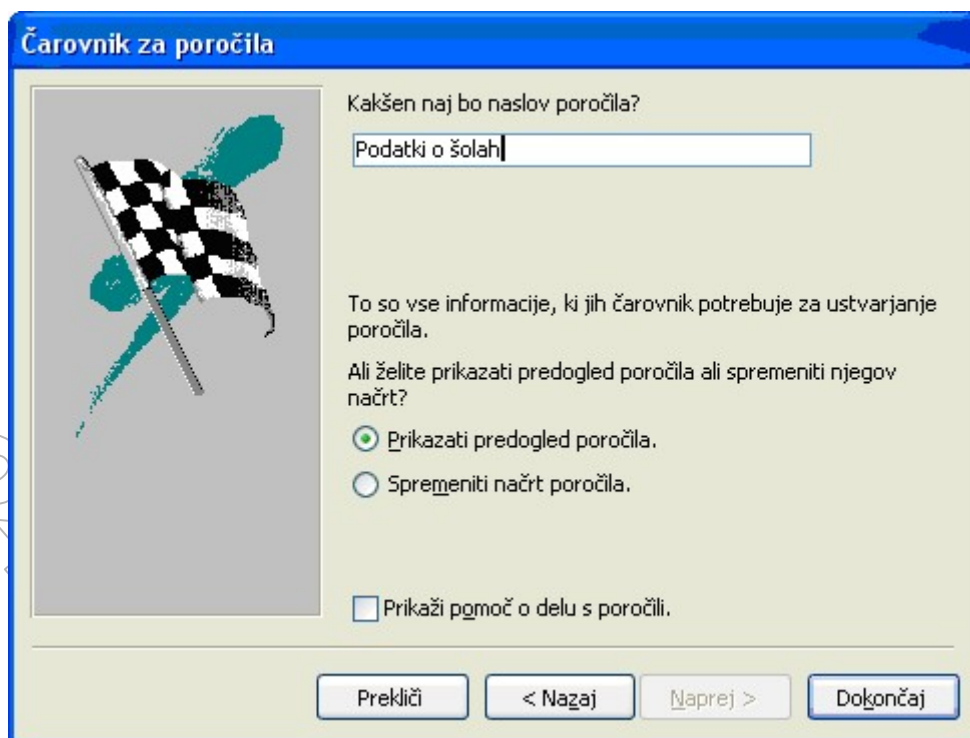
Slika 103 Izberemo kakšno postavitev poročila želimo

Zatem si moramo izbrati slog poročila. Mi smo se odločili za uraden slog (slika 104).



Slika 104 Izberemo slog poročila

Na koncu nas čarovnik vpraša še za naslov poročila (slika 105) in kaj želimo, da stori po izbiri dokončaj. Zahtevali smo, da nam naj prikaže poročilo. V primeru drugačne izbire pa bi prikazal načrt poročila, kjer lahko poročilo še prilagodimo svojim potrebam.



Slika 105 Vpišemo ime poročila

Izdelano poročilo je imelo nekoliko nerodno postavljene stolpce. Zato smo ponovno odprli poročilo v načinu načrta. Tam smo razširili polje za prikaz imena šole tako, da se prikazuje v dveh vrsticah. Končno poročilo vidimo na sliki 106.

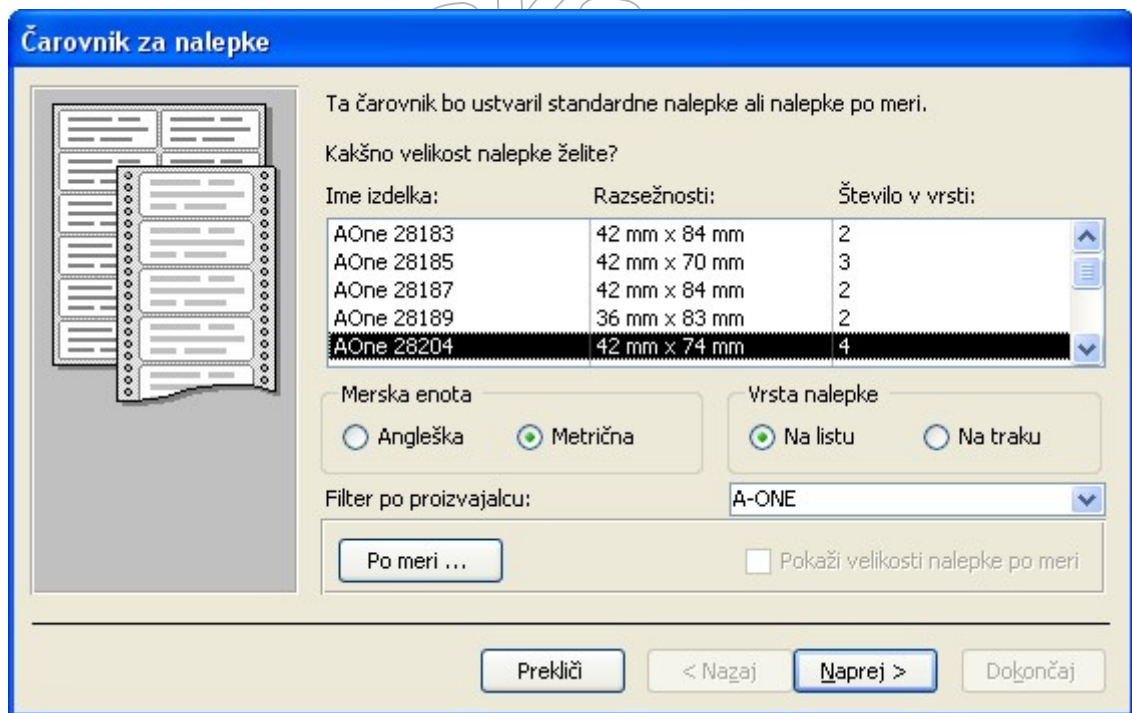
## Podatki o šolah

Šola	Polno ime	Naslov	Poštna številka
101	Biotekniški izobraževalni center Ljubljana - Gimnazija in veterinarska šola	Cesta v Mestni log 47	1000
102	Biotekniški izobraževalni center Ljubljana - Živilska šola	Lžanska c. 10	1000
103	Ekonomška šola Ljubljana	Prešernova 6	1000
104	Gimnazija Bežigrad	Pezičeva 4	1000
105	Gimnazija EURO šola Ljubljana	Litistrojska c. 40	1000
106	Gimnazija inekonomska srednja šola Trbovlje	Gimnazijska 10	1420
107	Gimnazija Jožeta Plečnika Ljubljana	Šubičeva 1	1000
201	Biotekniška šola Maribor	Koroška c. 13	2000

Slika 106 Poročilo s podatki o šolah

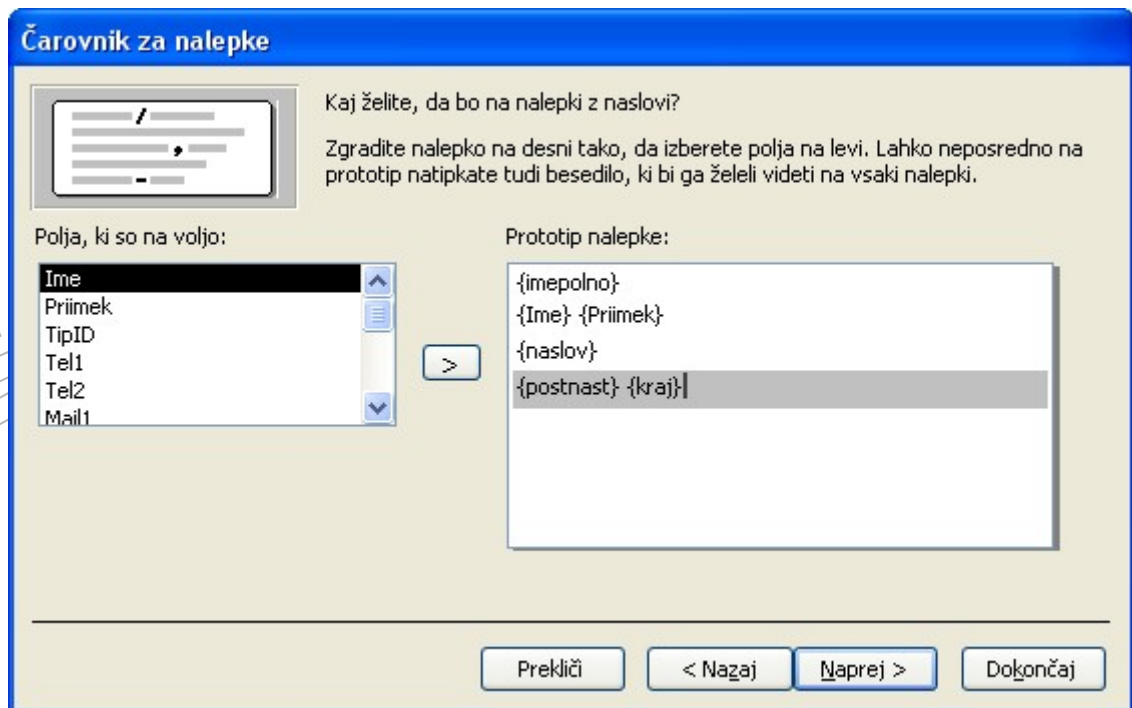
Na podoben način izdelamo tudi ostala poročila omenjena v poglavju 5.1.1.

Podrobneje si oglejmo še izdelavo nalepk. Kot vidimo na sliki 101, je med možnimi izbirami pri poročilih tudi »Čarovnik za nalepke«. Odpre se okno (slika 107), kjer izberemo velikost nalepk. Ustvarimo lahko nalepko po meri ( izberemo gumb »Po meri«) ali pa izberemo katero od ponujenih (npr. osenčena vrstica na sliki 107). Pri tem upoštevamo, na kakšne nalepke bomo poročilo dejansko tiskali.



Slika 107 Določimo vrsto nalepk

V naslednjem koraku (slika ni prikazana) določimo lastnosti in barvo pisave, ki bo uporabljena pri izpisu. Izberemo polja, ki jih želimo imeti na nalepki (slika 108) in jih ustrezno razporedimo.



Slika 108 Izberemo polja, ki jih želimo imeti na nalepki

Nalepke lahko izpišemo urejene po določenem polju. Na koncu tako kot za navadna poročila določimo še ime poročila (sliki nista prikazani). Dobimo poročilo, ki vsebuje izbrane



podatke postavljene tako, da so primerni za tiskanje na izbrano vrsto nalepk. Primer poročila vidimo na sliki 109.

Ne pozabimo omeniti, da smo tokrat za naše poročilo uporabili podatke na podlagi poizvedbe, ki smo si jo predhodno pripravili v ta namen. Tabela OSEBA namreč ne vsebuje naslovov in krajev. Zato smo pripravili poizvedbo, kjer smo podatke dopolnili še z ustreznimi podatki iz tabele SOLA in POSTA. Omejiti smo morali tudi izpis oseb le na koordinatorje in sicer tako, da smo izbrali le osebe tipa 2 in 3.

Biotehniški izobraževalni center Ljubljana - Živilska šola  
LIDIJA KOS  
Ižanska c. 10  
1000 Ljubljana

Biotehniški izobraževalni center Ljubljana - Živilska šola  
MAJA HRUST  
Ižanska c. 10  
1000 Ljubljana

Gimnazija EURO šola Ljubljana  
MIHA KRANJČ  
Litostrojska c. 40  
1000 Ljubljana

Gimnazija Jožeta Plečnika Ljubljana  
ŽIGA MAVER ZAJC  
Šubičeva 1  
1000 Ljubljana

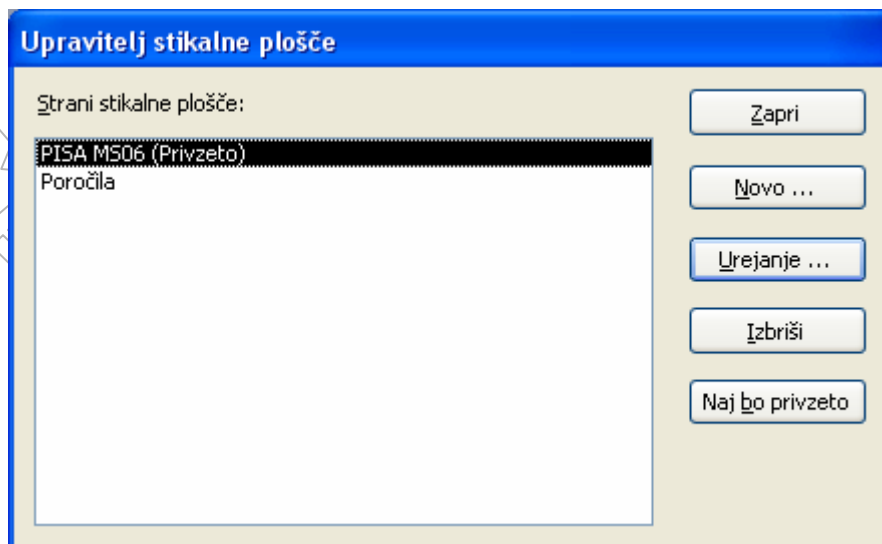
Slika 109 Poročilo v obliki nalepk

Možnosti za izdelavo poročil so res številne. Vsa izdelana poročila lahko shranimo. Natisnemo jih neposredno iz programa Access, možen pa je tudi izvoz poročil v Word.

## 5.6 Izdelava stikalne plošče

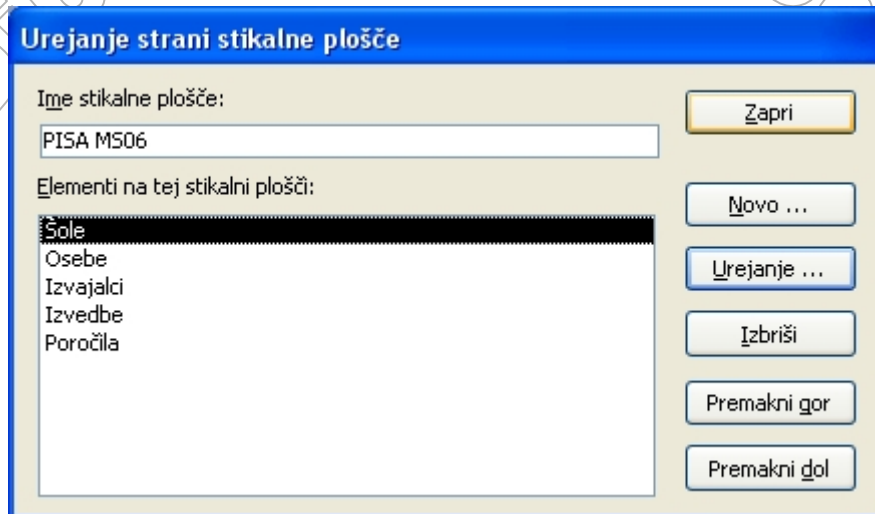
Vse izdelane obrazce in poročila je možno združiti v tako imenovano stikalno ploščo. Njen namen je predvsem olajšati delo uporabniku. Za vsak gumb na ekranu določimo določen dogodek, ki se izvede ob kliku na ta gumb. Poglejmo si, kako ploščo izdelamo. Ker želimo prikazati le osnovno funkcionalnost, se bomo namenoma omejili na zelo enostaven primer stikalne plošče. Izdelali bomo dve stikalni plošči in sicer PISA MS06 in Poročila.

V meniju Orodja v izbiri Orodja za zbirko podatkov izberemo Upravitelj stikalne plošče. Odpre se okno (slika 110). Gumb Novo nam omogoča izdelavo nove stikalne plošče. Vpisati je potrebno ime stikalne plošče, ki se zatem pojavi v levem oknu.



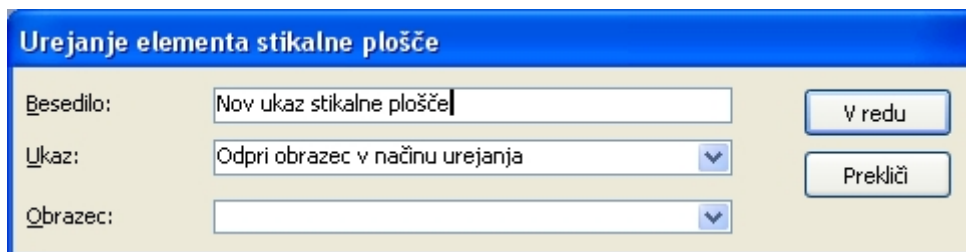
**Slika 110 Okno za izdelavo stikalne plošče**

Gumb »Urejanje« s slike 110 nam odpre okno (slika 111) v katerem dodajamo elemente na stikalno ploščo ali pa jih urejamo (jim spreminjamo lastnosti).



**Slika 111 Okno za urejanje stikalne plošče**

Ob izbiri Novo (na sliki 111) se odpre okno (slika 112), kjer določimo ime elementa ter ukaz, ki naj se izvrši ob kliku na ta element. V tretji vrstici izberemo ustrežno izbiro. Te se spreminjajo glede na izbrani ukaz.

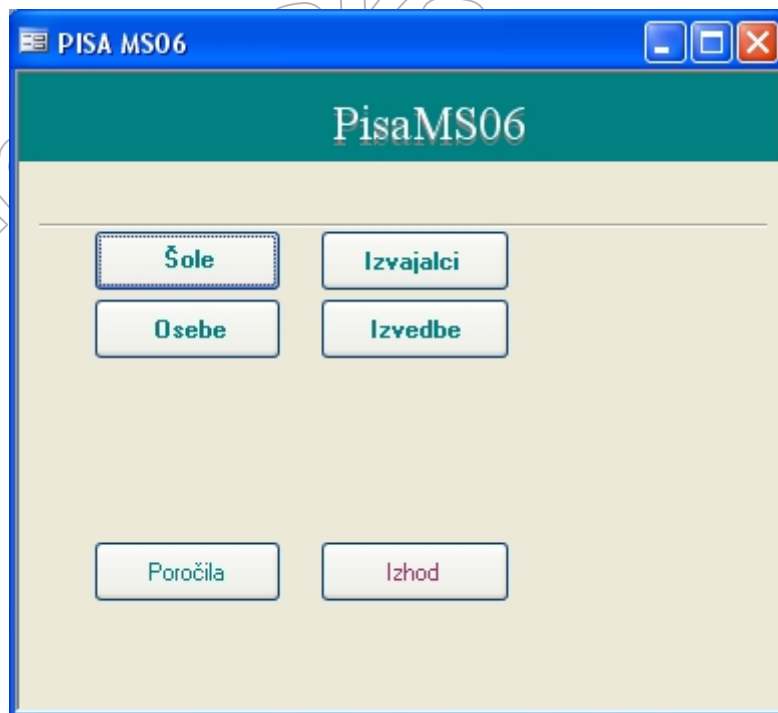


**Slika 112 Okno za urejanje elementa stikalne plošče**

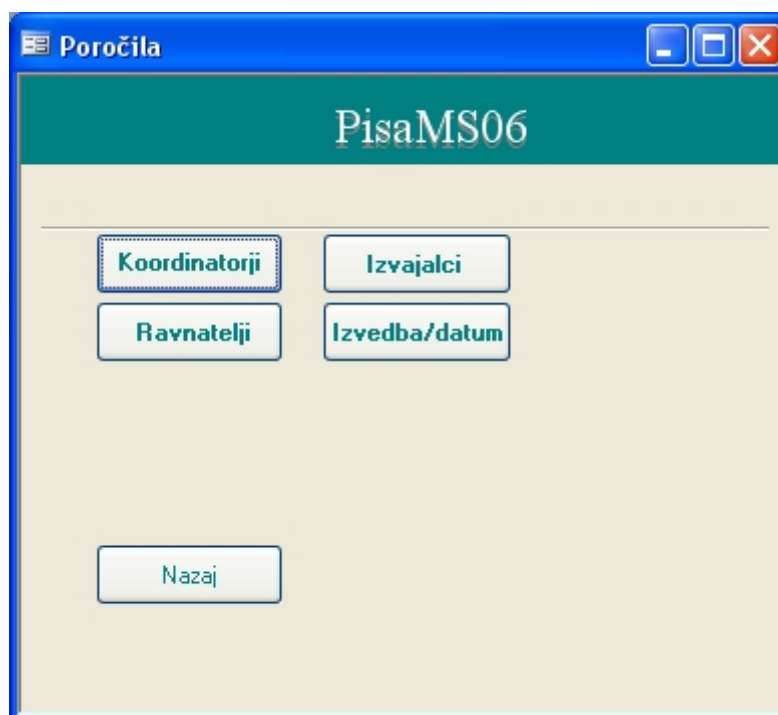
V primeru na sliki 112 nam tretja vrstica ponudi izbor enega od obrazcev, ki smo jih izdelali.

Ko smo na glavno stikalno ploščo dodali element Poročila, smo kot ukaz izbrali Pojdi v stikalno ploščo. Tokrat smo v tretji vrstici izbirali (pravzaprav je bila izbira le ena) še drugo, že prej izdelano, stikalno ploščo Poročila. Ko uporabnik klikne na gumb Poročila, torej preklopi na drugo stikalno ploščo.

Tudi stikalne plošče lahko v pogledu načrta podobno kot obrazce in poročila dodatno oblikujemo in prilagodimo našim potrebam. Tako na sliki 113 vidimo glavno stikalno ploščo PISA MS06. Iz te preko gumba Poročila lahko prestopimo v drugo stikalno ploščo Poročila, ki jo vidimo na sliki 114.



Slika 113 Glavna stikalna plošča



Slika 114 Stikalna plošča s poročili

Izdelali smo torej osnovno stikalno ploščo, s katero izbiramo različne možnosti dela s podatki v naši bazi. Na stikalno ploščo bi bilo možno dodati še številne elemente in dodatne funkcije, ki bi uporabniku poenostavile delo. Vendar tega v okviru diplomske naloge ne bomo obravnavali.

## 6 ZAKLJUČEK

V okviru diplomske naloge je izdelana podatkovna baza skupaj z osnovnimi obrazci in poročili. Sam projekt organizacije kontaktnih podatkov raziskave PISA bi lahko seveda razvijali še naprej. V nadaljevanju bi bila smiselna dopolnitev nekateri obrazcev ter poročil. Obrazce bi lahko oblikovno veliko bolj dodelali ali pa jih prilagodili izgledu mednarodne aplikacije, ki tudi deluje v programu Access. Na obrazce je mogoče dodati še dodatne kontrole in gumbe s funkcijami, kot je na primer gumb za tiskanje ali iskanje določenega zapisa. S pomočjo neposredno v Access vgrajenih možnosti ali s pomočjo programov, napisanih v programskem jeziku Microsoft Visual Basic for Applications, ki ga Access tudi podpira, bi lahko podrobno kontrolirali pravilnost podatkov, izdelovali kompleksna poročila, jih grafično obdelali in še mnogo več.

Popolnoma neizkoriščena je zaenkrat ostala tudi možnost povezave podatkov s spletom. Uporabnik trenutno tega ne potrebuje, bila pa bi mogoča dopolnitev aplikacije tudi v to smer.

Aplikacija bo z dodelavo stikalne plošče v precejšnji meri zadoščala trenutnim željam uporabnikov. V prihodnosti pa omogoča še številne dopolnitve in prilagoditve novim potrebam. Potrebne bodo tudi nastavitve za sočasno delo več uporabnikov.

## 7 LITERATURA

- [1] T. Mohorič, *Podatkovne baze*, BI-TIM, 2002
- [2] T. Mohorič, *Načrtovanje relacijskih podatkovnih baz*, BI-TIM, 1997
- [3] M. Gams, *Računalniški slovarček*, Cankarjeva založba, 1993
- [4] P. Baloh, P. Vrečar, *Ob praktičnih primerih skozi MS Office: vodnik za preživetje na delovnem mestu. Dopolnjena izdaja*. Podplat; Velenje: samozal., 2002
- [5] J. Celko, *Joe Celko's SQL for smarties: Advanced SQL programming*, Morgan Kaufmann Publishers, Inc., San Francisco, California, 1995
- [6] <http://www.functionx.com/access/>, dostop: 1. 10. 2006