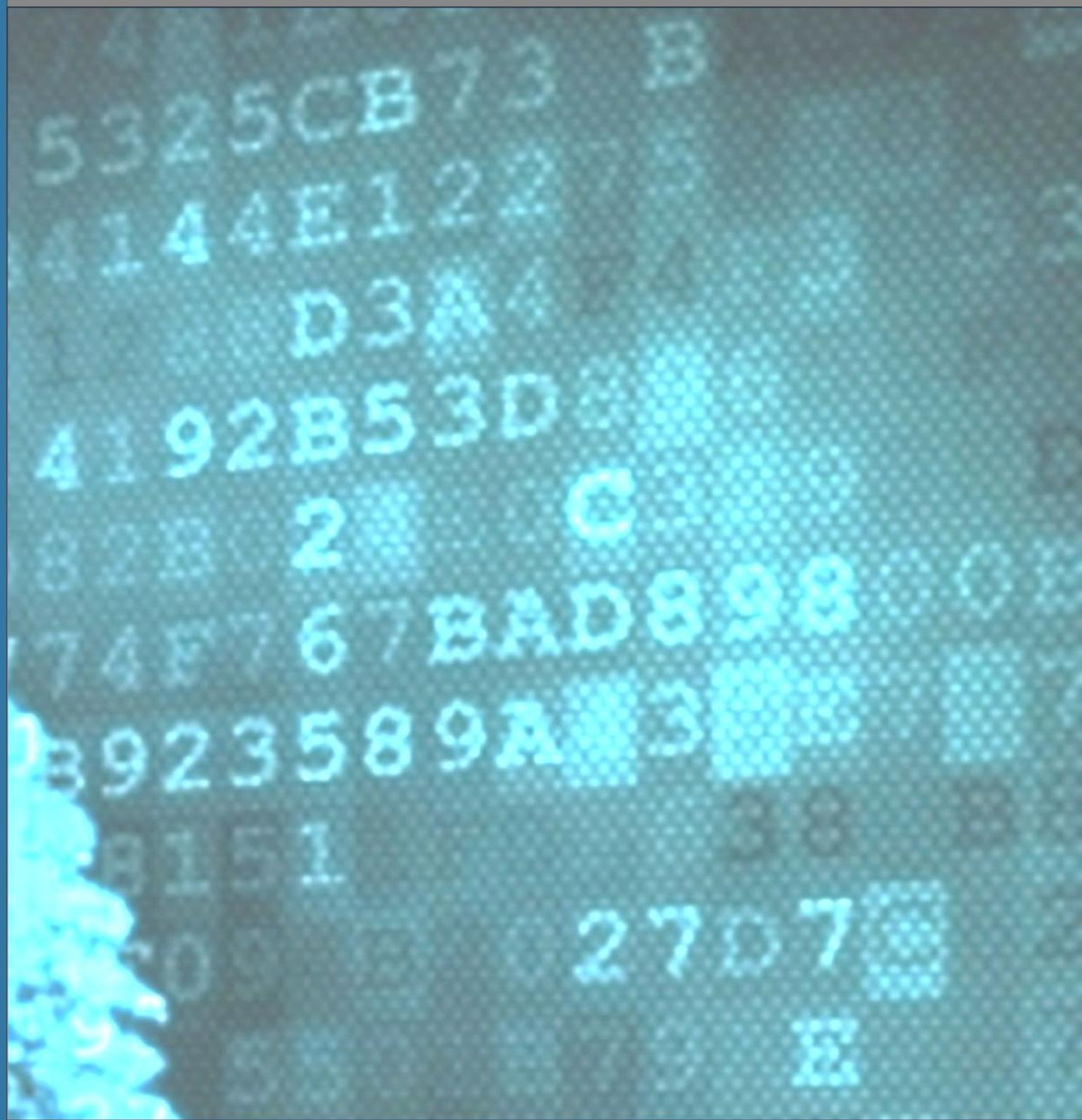




**Damir Vuk**  
**Enes Ciriković**

## Priručnik za laboratorijske vježbe iz baza podataka



© Damir Vuk i Enes Ciriković, 2015.

**Recezent**

dr. sc. Oliver Jukić, prof. v. š.  
mr. sc. Željko Knok, v. pred.

**Izdavač:**

Visoka škola za menadžment u turizmu i informatici u Virovitici,  
Matije Gupca 78, 33000, Virovitica

<http://www.vsmti.hr>

**Za izdavača:**

Doc.dr.sc.Vesna Bedeković, prof.v.š.

**Grafička priprema:**

Ivan Heđi, dipl. ing.

**Lektura:**

Ivana Vidak, prof.

Ovaj materijal predstavlja digitalni priručnik za laboratorijske vježbe iz kolegija Baze podataka. Materijal je besplatno dostupan na web stranicama Visoke škole za menadžment u turizmu i informatici u Virovitici odnosno na službenim stranicama kolegija Baze podataka. Odluku o odobrenju elektroničke publikacije „Priručnik za laboratorijske vježbe iz Baza podataka – elektronička verzija“ (Klasa: 612-10/15-03/06) donijelo je Stručno vijeće Visoke škole za menadžment u turizmu i informatici u Virovitici na svojoj sjednici od 16. 06. 2015. Na istoj sjednici donešena je odluka o izmjenama i dopunama u studijskim programima Visoke škole za menadžment u turizmu i informatici u Virovitici (Klasa: 612-10/15-03/06, Urbroj: 2189-74-15/06) kojom se ova publikacija uvrštava kao obvezna literatura na kolegiju „Baze podataka“ koji se izvodi u petom semestru studija Menadžment, smjer Informatički menadžment.

ISBN 978-953-8028-03-8

**Damir Vuk**  
**Enes Ciriković**

**PRIRUČNIK ZA LABORATORIJSKE  
VJEŽBE IZ  
BAZA PODATAKA**

**ELEKTRONIČKA VERZIJA  
(1.0)**

Virovitica, 2015.



# I. Predgovor

Područje baza podataka je kako u teorijskom, tako i u praktičnom smislu vrlo opsežno, kompleksno i diverzificirano. Iako je teorija baza podataka relativno dobro fundirana, postoji velik raskorak između teorijskog pristupa te konkretnih alata i njihove primjene. Jedina konzistentna teorija kao fundacija za razvoj baza podataka je relacijska teorija poznata kao „Teorija relacijskog modela podataka“, čiji začeci leže u poznatom radu E.F. Coda: „[A relational model of data for large shared data banks](#)“ koji datira još iz 1970. godine<sup>1</sup>. Ta je teorija stvorila plodno tlo za cijeli niz radova koji su nadopunjavali njene pojedine aspekte. I sam Codd je u više svojih naknadnih radova, pojašnjavao dileme u vezi izvornog relacijskog modela, postavljajući niz eksplicitnih pravila u vezi potrebne vjernosti SUBP<sup>2</sup> s relacijskim modelom. Iako se danas često govori o različitim modelima podataka, niti jedan od njih nije tako čvrsto teorijski zasnovan kao što je to relacijski model. Neki od njih zapravo i nisu modeli podataka u pravom smislu, jer ne samo da nisu teorijski dobro zasnovani, nego su i nekompletni.

Ideja relacijskog modela je bila da se na logičkoj razini definiraju principi za dobru organizaciju podataka u bazi podataka. Osobito značajan je doprinos relacijskog modela u konceptu da se razdvoji logička od fizičke razine organizacije podataka. Time se postiže fizička neovisnost podataka. Dotadašnji koncepti baza podataka su imali čvrsto povezanu logičku i fizičku razinu, što je izazivalo cijeli niz problema. Relacijska teorija je nastala upravo s ciljem da se ti problemi riješe na teorijskoj razini, a zatim da se prakticiraju kako u oblikovanju baza podataka, tako i u postavljanju arhitekture sustava za upravljanje bazom podataka.

Paradoksalno je da relacijski model podataka ni do danas nije implementiran u niti jedan sustav za upravljanje bazama podataka na način da bude potpuno u skladu s teorijom relacijskog modela podataka. Za to postoji više razloga. Prvi problem je teorijske prirode. On proizlazi iz načina na koji je relacijska teorija podataka shvaćana. Drugi problem proizlazi iz načina na koji je relacijski model podataka implementiran u SUBP te posljedično tome, komercijalnih interesa proizvođača takvih sustava.

U vrijeme svoga nastanka, teorija relacijskog modela je bila radikalno nova i nedovoljno shvaćena. S tim u vezi, poznata je stručna i akademska rasprava o mjerodavnosti te teorije kao teorijske podloge za razvoj baza podataka odnosno sustava za upravljanje bazama podataka, koja je kulminirala sredinom sedamdesetih godina prošlog stoljeća. Ta višegodišnja rasprava nazvana „Great debate“<sup>3</sup> kulminirala je donošenjem ANSI/X3/SPARC preporuke o trirazinskoj shemi baze podataka (konceptualna, eksterna i

---

<sup>1</sup> Codd, E. F. (1970). *A relational model of data for large shared data banks*. *Communications of the ACM*, 13(6), 377-387.

<sup>2</sup> SUBP je skraćenica od Sustav za upravljanje bazom podataka (engl: DBMS – Database management system)

<sup>3</sup> Tsichritzis, D., & Klug, A. (1978). *The ANSI/X3/SPARC DBMS framework report of the study group on database management systems*. *Information systems*, 3(3), 173-191. *management systems*. *Information systems*, 3(3), 173-191.

fizička), čime je razdvojena logička (konceptualna i eksterna) od fizičke sheme. Ubrzo nakon toga rasprava se proširila na semantičku snagu modela podataka, s tvrdnjom da je semantička izražajnost relacijskog modela nedovoljna. Do sredine osamdesetih godina je razvijeno više tzv. semantičkih modela. Osamdesetih godina se iz tog koncepta razvio koncept objektnih modela podataka i ideja o tome da će objektni model zamijeniti relacijski, odnosno da će relacijske baze i SUBP biti zamijenjeni objektnim bazama, što se nije dogodilo. U devedesetim godinama su se pojavila dva nova koncepta koja su tumačena kao korak naprijed u razvoju baza podataka. Prvi je bio koncept skladišta podatka. Drugi je bio koncept tzv. postrelacijskih baza podataka odnosno objektno relacijske baze podatka. Dok koncept skladišta podataka ne predstavlja negaciju relacijske teorije, koncept objektno-relacijskih baza podataka zapravo nema ništa s konzistentnom teorijom – radi se o marketinškom triku velikih proizvođača SUBP. U proteklih desetak godina pojavljuju se nova dva povezana koncepta koji najavljuju tzv. „novo doba baza podataka“. To su koncepti NoSQL baza i koncept Big Data. Zajednička karakteristika svih tih inicijativa je da se ne zasnivaju na jedinstvenoj i konzistentnoj teoriji te da su njihove implementacije usko ograničene u aplikativnom smislu. Nadalje, kod NoSQL i Big Data, radi se specijalnom slučaju baza podataka odnosno repozitoriju podataka, što nije nikako zamjena za relacijske baze podataka.

Prve implementacije SUBP su bile skromne u pogledu vjernosti s relacijskom teorijom. Proizvođači SUBP nisu htjeli žrtvovati performanse brzine u odnosu na postojeće SUBP prisutne na tržištu. Budući da je u to vrijeme cjelokupna računalna snaga bila izrazito skromna, rješenje je bilo da se implementiraju samo minimalne osobine relacijskog modela, ali da se pritom agresivno marketinški stvara predodžba kod kupaca kako su njihovi proizvodi u skladu s relacijskim modelom baza podataka, jednostavno zato jer je to isto nudila konkurencija. Osobito negativna posljedica na razvoj SUBP je proizašla je iz činjenice da prva verzija službenog ANSI standarda za SQL jezik iz 1987 godine nije bila zasnovana na relacijskoj teoriji, nego je bila gotovo u potpunosti preuzeta iz jedne takve implementacije korporacije IBM<sup>4</sup>. Iduće verzije SQL standarda su još više pogoršavale tu neusklađenost SQL-a i teorijskog modela. Glavni uzrok je proizlazio iz činjenice da je među članovima komiteta za SQL standard prevlast imao lobi proizvođača SUBP, koji su uzajamno uvažavali vlastite komercijalne interese.

Posljedica svega toga je stanje u kojem imamo dobru staru teoriju (relacijski model podataka) te sustave za upravljanje bazama podataka koji su zahvaljujući SQL standardu kojega također samo djelomično podržavaju, u ozbiljnom neslaganju u odnosu na teoriju. Ovdje je veći problem u odnosu na osobine koji ti sustavi podržavaju a nisu predviđene niti teorijom, a niti SQL standardom, nego u odnosu na osobine koje ti sustavi ne podržavaju, a relacijska teorija i SQL standard ih zahtijevaju.

Još lošija posljedica je činjenica da na temelju navedenog u svijetu imamo mnogo profesionalaca baza podataka koji svoju karijeru zasnivaju na „teorijskim znanjima“ koja proizlaze iz alata (SUBP), a ne na temelju prave konzistentne teorije. Nadalje, mnogi udžbenici i mnogi tečajevi su sačinjeni u stilu „kuharica“ (engl. Cookbook) koje nemaju jasnu teorijsku podlogu. Rezultat korištenja takvih izvora u

---

<sup>4</sup> Date, C. J., & Darwen, H. (1997). *A Guide To Sql Standard (Vol. 3)*. Reading: Addison-Wesley.

učenju o bazama podataka neminovno dovodi do negativne posljedice da su profesionalci istrenirani na temelju alata, s nedovoljnim uvidom u „širu sliku“ područja s kojim se bave<sup>5</sup>. U moru konzumerizma i globalne moći korporacija niti područje baza podataka nije otok. Autor relacijskog modela Codd je umro i praktično više od dvadeset godina nema neposredni utjecaj na stručnu i akademsku javnost. Danas je najagilniji zagovornik Coddove teorije njegov bliski suradnik iz sedamdesetih godina Criss Date, inače autor jednog od najpoznatijih sveučilišnih udžbenika iz područja baza podataka „An Introduction to Database Systems“<sup>6</sup> izdanog u gotovo milijun primjeraka u devet izdanja. S nekolicinom teorijskih istomišljenika on i danas djeluje na stalnom unapređivanju i širenju izvorne ideje relacijske teorije baza podataka<sup>7</sup>.

Napravili smo ovako dugi uvod, kako bismo naglasili i ukratko opisali kontekst u kome se danas podučavanje baza podataka nalazi te istakli na što pritom treba obratiti pažnju.

Ovaj priručnik za laboratorijske vježbe iz baza podataka nije glavni udžbenik za baze podataka. On predstavlja dopunski, ali ne manje važan dio poučavanja baza podataka. Pritom se naglašava pored praktične strane, također i njegova teorijska zasnovanost. On se temelji na sljedećim idejama:

- Da se kroz jednostavne praktične primjere potpomogne praćenje teorijske nastave iz baza podataka ( predavanja), imajući u vidu da je studentima je često teško pratiti teorijsku nastavu jer nemaju dovoljnu predodžbu o tome što je to baza podataka.
- Da se potpomogne učenje i razumijevanje izložene teorijske nastave. – Nakon što studenti slušaju teorijsku nastavu oni trebaju na konkretnim alatima praktično utvrditi/potvrditi svoje znanje.
- Da se studente kroz nastavni proces vježbi upozna s minimalnim, ali često vrlo bitnim elementima u savladavanju nekih praktičnih alata, kako bi studenti na taj način dobili osnovu za samostalno proučavanje – barem oni koji za to imaju sklonosti odnosno interesa.
- Da se studente uključi u aktivno izvršavanje praktičnih zadataka. – Mnogi studenti teže pasivnom učenju – kroz ove vježbe od studenata se očekuje aktivno učenje u pripremi vježbe, radu u laboratoriju, te naknadni rad kod kuće (dovršetak vježbe i produbljanje znanja).
- Da se studente stimulira na aktivnost kroz bodovanje te aktivnosti kao udjela u ukupnom rezultatu ispita. To se izvodi putem blic-testova (putem e-learnig sistema) kojima se provjerava pripremljenost za vježbu te stupanj savladavanja vježbe, odnosno aktivnost tijekom vježbe.

Ovaj priručnik je nastao na osnovu promišljanja, ali isto tako i praktičnog iskustva u kompjutorskom labu s više generacija studenata. Kao što smo u uvodu naveli, baze podataka su opsežno i kompleksno područje. U 14 dvosatnih laboratorijskih vježbi nije moguće pokriti svu tematiku baza podataka. Više važnih tema kao što su npr., transakcije i procedure, trigeri, DCL komande, složeni

---

<sup>5</sup> Morien, R. I. (2006). *A Critical Evaluation Database Textbooks, Curriculum and Educational Outcomes. Director, 07*

<sup>6</sup> Date, C. J. (2003). *An Introduction to Database Systems, Eighth Edition, Addison-Wesley, 2003. ISBN 0-321-19784-4.*

<sup>7</sup> Njihovi stavovi i rasprave se mogu pronaći na web-adresi: <http://www.dbdebunk.com/>

agregati i drugo, nisu obrađeni. Na temelju iskustva s dosadašnjim grupama studenata smatramo da je opseg vježbi i zadataka koji se od studenata očekuje da izvrše (prema ovom priručniku), na granici maksimalno prihvatljivog opterećenja, te ga ni u kom slučaju ne bi trebalo proširiti, jer bi to moglo biti kontraproduktivno. Također, oblikovanje baza podataka koje obuhvaća Entity-Relationship modeliranje i normalizaciju, nije dio ovih vježbi, budući da se izvodi u okviru konzultativne nastave i rada na samostalnim projektima. Ti ovdje neobrađeni dijelovi baza podataka spadaju u napredniji kurs (kolegij), pa bi ga po našem mišljenju trebalo obraditi i izučavati odvojeno od ovdje izloženoga, uvoda u baze podataka.

Uzimajući sve gore navedeno, za praktični rad odnosno izvođenje vježbi, potrebno je odabrati pogodne softverske alate. Prvi među njima je Microsoft Access 2010. Iskustvo govori da je Access za to vrlo pogodan alat jer :

- Omogućuje da studenti u njemu dođu do rezultata brzo i uz minimalni napor,
- Jednostavno se uči, posjeduje niz automatskih generatora i vizualni razvoj,
- Studenti koji nemaju dovoljno znanja vrlo brzo savladavaju elementarnu upotrebu,
- Posjeduje dobru integraciju baze podataka i dodatnih razvojnih alata,
- DBMS – engine je dovoljno relacijski zasnovan pa se može koristiti za nastavu,
- Studenti mogu znanja stečena u radu s Accessom, s relativnom lakoćom prenijeti na druge razvojne alate i SUBP.

Koliko nam je poznato niti jedan razvojni alat za područje baza podataka ne posjeduje sva ta navedena svojstva. Pored Accessa koristi se i ODBC/SQL Server, te FirebirdSQL i MySQL. Konceptija Accessa već od njegovih najranijih verzija, je upravo takva da se prirodno povezuje i nadograđuje na SQL Server. FirebirdSQL odlikuje niz dobrih osobina koje ga čine pogodnim za upotrebu: besplatan je, kompaktan je, podržava SQL standard. MySQL je najraširenija SUBP te unatoč nekih nedostataka, predstavlja korisno iskustvo za studente, pogotovo kad se uzme u obzir da je to jedan od nakorištenijih SUBP u Web aplikacijama.

Ovaj priručnik je namijenjen kao obavezni priručnik studentima za laboratorijske vježbe iz kolegija baze podataka, koji se izvode na studiju informatičkog menadžmenta u Visokoj školi za menadžment u turizmu i informatici u Virovitici. Priručnik je postupno nastajao, a isto tako se i dalje nastavlja razvijati u skladu s budućim iskustvom te raspoloživim resursima i razvojem informacijske tehnologije. Stoga su sve buduće primjedbe kako kolega nastavnika, tako i studenata, dobro došle.

U Virovitici, srpanj 2015.

Autori:

[mr. sc. Damir Vuk, v.pred.](#)

[Enes Ciriković, dipl.ing.](#)





## II. Sadržaj

I.	PREGOVOR.....	1
II.	SADRŽAJ .....	6
III.	POPIS VJEŽBI.....	11
IV.	UPUTE ZA KORIŠTENJE PRIRUČNIKA .....	12
V.	LITERATURA .....	13
<b>VJEŽBA 1:</b>	<b>UVOD U MICROSOFT ACCESS  </b> .....	<b>15</b>
1.1	MOTIVACIJA .....	15
1.2	CILJEVI VJEŽBE – ISHODI UČENJA.....	16
1.3	SAMOSTALNA PRIPREMA VJEŽBE.....	16
1.4	TEORIJSKI DIO PRIPREME VJEŽBE .....	17
1.4.1	<i>Microsoft Access – uvod</i> .....	17
1.4.2	<i>Razvoj Ms Access-a</i> .....	18
1.4.3	<i>Arhitektura Ms Accessa</i> .....	18
1.4.4	<i>Komponente Ms Accessa</i> .....	20
1.5	SADRŽAJ I ZADATAK VJEŽBE .....	21
1.5.1	<i>Izbornici i alatne trake</i> .....	21
1.5.2	<i>Aplikacija „Northwind Traders“</i> .....	24
1.5.3	<i>Tijek izvođenja vježbe</i> .....	27
1.6	ZAVRŠNA PITANJA I ZADACI .....	28
1.6.1	<i>Završna pitanja u vezi izložene materije vježbe</i> .....	28
1.6.2	<i>Zadaci za dodatno samostalno produblivanje znanja</i> .....	29
1.7	LITERATURA I DODATNI IZVORI .....	29
<b>VJEŽBA 2:</b>	<b>KREIRANJE TABLICA U MSACCESSU  </b> .....	<b>31</b>
2.1	MOTIVACIJA .....	31
2.2	CILJEVI VJEŽBE – ISHODI UČENJA.....	32
2.3	SAMOSTALNA PRIPREMA VJEŽBE.....	32
2.4	TEORIJSKI DIO PRIPREME VJEŽBE .....	33
2.4.1	<i>Tablice u Ms Access-u</i> .....	33
2.4.2	<i>Tipovi podataka</i> .....	34
2.4.3	<i>Ključevi relacije</i> .....	35
2.4.4	<i>Ograničenja i formati na razini atributa</i> .....	37
2.5	SADRŽAJ I TIJEK IZVOĐENJA VJEŽBE.....	38
2.5.1	<i>Sadržaj vježbe</i> .....	38
2.5.2	<i>Tijek izvođenja vježbe</i> .....	39
2.6	ZAVRŠNA PITANJA I ZADACI .....	39
2.6.1	<i>Završna pitanja</i> .....	39
2.6.2	<i>Zadaci za samostalno produblivanje znanja</i> .....	40
2.7	LITERATURA I DODATNI MATERIJALI.....	40
<b>VJEŽBA 3:</b>	<b>QUERY, POGLEDI, OGRANIČENJA  </b> .....	<b>41</b>
3.1	MOTIVACIJA .....	41
3.2	CILJEVI VJEŽBE – ISHODI UČENJA.....	42
3.3	SAMOSTALNA PRIPREMA VJEŽBE.....	42
3.4	TEORIJSKA PRIPREMA VJEŽBE .....	43
3.4.1	<i>Dodatna ograničenja atributa</i> .....	43
3.4.2	<i>Strani ključ</i> .....	45
3.4.3	<i>Uspostavljanje referencijalne veze</i> .....	46
3.4.4	<i>QBE - Query</i> .....	48
3.4.5	<i>Validacijski pogledi</i> .....	49
3.5	ZADACI I SADRŽAJ VJEŽBE .....	51
3.5.1	<i>Sadržaj vježbe</i> .....	51
3.5.2	<i>Tijek izvođenja vježbe</i> .....	52
3.6	ZAVRŠNA PITANJA .....	53
3.7	ZADACI ZA SAMOSTALNO PRODUBLIVANJE ZNANJA .....	54
3.8	LITERATURA I DODATNI MATERIJALI.....	54

<b>VJEŽBA 4:</b>	<b>PROGRAMSKO SUČELJE BP – FORME</b>	<b>55</b>
4.1	MOTIVACIJA .....	55
4.2	CILJEVI VJEŽBE – ISHODI UČENJA.....	56
4.3	SAMOSTALNA PRIPREMA VJEŽBE.....	56
4.4	TEORIJSKA PRIPREMA VJEŽBE .....	57
4.4.1	<i>Semantika u bazi podataka</i> .....	57
4.4.2	<i>DBMS/SUBP</i> .....	58
4.4.3	<i>Programsko sučelje prema bazi podataka</i> .....	58
4.4.4	<i>Ekranse forme</i> .....	59
4.4.5	<i>Vrste formi</i> .....	62
4.4.6	<i>Dodavanje podatkovnih objekata</i> .....	62
4.4.7	<i>Dodavanje ostalih važnijih objekata</i> .....	63
4.5	ZADACI I SADRŽAJ VJEŽBE .....	65
4.5.1	<i>Sadržaj vježbe</i> .....	65
4.5.2	<i>Tijek izvođenja vježbe</i> .....	65
4.6	ZAVRŠNA PITANJA I ZADACI .....	66
4.6.1	<i>Pitanja u vezi izložene vježbe</i> .....	66
4.6.2	<i>Zadaci za samostalno produblivanje znanja</i> .....	66
4.7	LITERATURA I DODATNI MATERIJALI.....	66
<b>VJEŽBA 5:</b>	<b>PROGRAMSKO SUČELJE BP – SUBFORME I IZVJEŠTAJI</b>	<b>67</b>
5.1	MOTIVACIJA .....	67
5.2	CILJEVI VJEŽBE – ISHODI UČENJA.....	68
5.3	SAMOSTALNA PRIPREMA VJEŽBE.....	68
5.4	TEORIJSKA PRIPREMA VJEŽBE .....	69
5.4.1	<i>Vrste povezivanja podataka s objektima forme</i> .....	69
5.4.2	<i>Subforme</i> .....	70
5.4.3	<i>Izveštaji</i> .....	72
5.4.4	<i>Struktura izvještaja</i> .....	73
5.4.5	<i>Kreiranje izvještaja</i> .....	74
5.5	ZADACI I SADRŽAJ VJEŽBE .....	78
5.5.1	<i>Sadržaj vježbe</i> .....	78
5.5.2	<i>Tijek izvođenja vježbe</i> .....	78
5.6	ZAKLJUČNA PITANJA I ZADACI.....	79
5.6.1	<i>Pitanja u vezi izložene vježbe</i> .....	79
5.6.2	<i>Zadaci za samostalno produblivanje znanja</i> .....	79
5.7	LITERATURA I DODATNI MATERIJALI.....	80
<b>VJEŽBA 6:</b>	<b>SQL KOMANDE – DDL</b>	<b>81</b>
6.1	MOTIVACIJA .....	81
6.2	CILJEVI VJEŽBE – ISHODI UČENJA.....	82
6.3	SAMOSTALNA PRIPREMA VJEŽBE.....	82
6.4	TEORIJSKA PRIPREMA VJEŽBE .....	83
6.4.1	<i>SQL – jezik baza podataka</i> .....	83
6.4.2	<i>Elementi SQL komande</i> .....	84
6.4.3	<i>SQL DDL</i> .....	84
6.4.4	<i>CREATE TABLE komanda</i> .....	85
6.4.5	<i>ALTER TABLE komanda</i> .....	86
6.4.6	<i>CONSTRAINT klauzula</i> .....	87
6.4.7	<i>DROP komanda/klauzula</i> .....	88
6.4.8	<i>Kreiranje SQL DDL komandi u Accessu</i> .....	89
6.5	ZADATAK I SADRŽAJ VJEŽBE .....	91
6.5.1	<i>Sadržaj vježbe</i> .....	91
6.5.2	<i>Tijek izvođenja vježbe</i> .....	91
6.6	ZAKLJUČNA PITANJA I DODATNI ZADACI.....	92
6.6.1	<i>Zadaci za samostalno produblivanje znanja</i> .....	93
6.7	LITERATURA I DODATNI MATERIJALI.....	93
<b>VJEŽBA 7:</b>	<b>ODBC</b>	<b>95</b>
7.1	MOTIVACIJA .....	95
7.2	CILJEVI VJEŽBE – ISHODI UČENJA.....	96

7.3	SAMOSTALNA PRIPREMA VJEŽBE.....	96
7.4	TEORIJSKI DIO PRIPREME VJEŽBE.....	97
7.4.1	ODBC – Open Database Connectivity.....	97
7.4.2	ODBC arhitektura.....	98
7.4.3	Arhitektura Ms Accessa.....	98
7.4.4	ODBC veza s SQL Server SUBP.....	100
7.5	SADRŽAJ I ZADATAK VJEŽBE.....	105
7.5.1	Tijek izvođenja vježbe.....	105
7.6	ZAVRŠNA PITANJA I ZADACI.....	106
7.6.1	Završna pitanja u vezi izložene materije vježbe.....	106
7.6.2	Zadaci za dodatno samostalno produblivanje znanja.....	106
7.7	LITERATURA I DODATNI IZVORI.....	106
<b>VJEŽBA 8:</b>	<b>SQL DML</b>   .....	<b>107</b>
8.1	MOTIVACIJA.....	107
8.2	CILJEVI VJEŽBE – ISHODI UČENJA.....	108
8.3	SAMOSTALNA PRIPREMA VJEŽBE.....	108
8.4	TEORIJSKI DIO PRIPREME VJEŽBE.....	109
8.4.1	Komanda SELECT.....	109
8.4.2	Komanda UNION.....	113
8.4.3	Komanda UPDATE.....	114
8.4.4	Komanda DELETE.....	114
8.4.5	Komanda INSERT INTO.....	115
8.4.6	Komanda SELECT INTO.....	116
8.4.7	Kreiranje DML komandi u Access-u.....	117
8.5	SADRŽAJ I ZADATAK VJEŽBE.....	118
8.5.1	Tijek izvođenja vježbe.....	118
8.6	ZAVRŠNA PITANJA I ZADACI.....	120
8.6.1	Završna pitanja u vezi izložene materije vježbe.....	120
8.7	LITERATURA I DODATNI IZVORI.....	120
<b>VJEŽBA 9:</b>	<b>SQL SPOJEVI</b>   .....	<b>121</b>
9.1	MOTIVACIJA.....	121
9.2	CILJEVI VJEŽBE – ISHODI UČENJA.....	122
9.3	SAMOSTALNA PRIPREMA VJEŽBE.....	122
9.4	TEORIJSKI DIO PRIPREME VJEŽBE.....	123
9.4.1	SQL spojevi.....	123
9.4.2	Unutarnji spoj (INNER JOIN).....	123
9.4.3	Vanjski spoj (OUTER JOIN).....	124
9.5	SADRŽAJ I ZADATAK VJEŽBE.....	125
9.5.1	Tijek izvođenja vježbe.....	125
9.6	ZAVRŠNA PITANJA I ZADACI.....	126
9.6.1	Završna pitanja u vezi izložene materije vježbe.....	126
9.7	LITERATURA I DODATNI IZVORI.....	126
<b>VJEŽBA 10:</b>	<b>FIREBIRD – INSTALACIJA I UPOTREBA</b>   .....	<b>127</b>
10.1	MOTIVACIJA.....	127
10.2	CILJEVI VJEŽBE – ISHODI UČENJA.....	128
10.3	SAMOSTALNA PRIPREMA VJEŽBE.....	128
10.4	TEORIJSKI DIO PRIPREME VJEŽBE.....	129
10.4.1	Uvod u Firebird RDBMS.....	129
10.4.2	Firebird instalacija.....	130
10.4.3	Vizualni alati za rad s Firebird relacijskom bazom podataka.....	131
10.4.4	Firebird ODBC driver.....	134
10.5	SADRŽAJ I ZADATAK VJEŽBE.....	135
10.5.1	Tijek izvođenja vježbe.....	135
10.6	ZAVRŠNA PITANJA I ZADACI.....	136
10.7	LITERATURA I DODATNI IZVORI.....	136
<b>VJEŽBA 11:</b>	<b>SQL OPERATORI I NULL VRIJEDNOST</b>   .....	<b>137</b>
11.1	MOTIVACIJA.....	137
11.2	CILJEVI VJEŽBE – ISHODI UČENJA.....	138

11.3	SAMOSTALNA PRIPREMA VJEŽBE.....	138
11.4	TEORIJSKI DIO PRIPREME VJEŽBE.....	139
11.4.1	<i>SQL operatori</i> .....	139
11.4.2	<i>NULL vrijednost</i> .....	141
11.5	SADRŽAJ I TIJEK IZVOĐENJA VJEŽBE.....	143
11.5.1	<i>Sadržaj vježbe</i> .....	143
11.5.2	<i>Tijek izvođenja vježbe</i> .....	143
11.6	ZAVRŠNA PITANJA I ZADACI.....	145
11.6.1	<i>Završna pitanja</i> .....	145
11.6.2	<i>Zadaci za samostalno produblivanje znanja</i> .....	145
11.7	LITERATURA I DODATNI MATERIJALI.....	146
<b>VJEŽBA 12: SQL AGREGACIJE, SORTIRANJE   ..... 147</b>		
12.1	MOTIVACIJA.....	147
12.2	CILJEVI VJEŽBE – ISHODI UČENJA.....	148
12.3	SAMOSTALNA PRIPREMA VJEŽBE.....	148
12.4	TEORIJSKI DIO PRIPREME VJEŽBE.....	149
12.4.1	<i>Agregatne funkcije u SQL-u</i> .....	149
12.4.1.1	Funkcija COUNT.....	150
12.4.1.2	Agregatne funkcije MAX i MIN.....	150
12.4.1.3	Agregatne funkcije SUM i AVG.....	150
12.4.2	<i>Klauzula GROUP BY</i> .....	151
12.4.3	<i>Klauzula HAVING</i> .....	152
12.4.4	<i>Klauzula ORDER BY</i> .....	153
12.5	SADRŽAJ I ZADATAK VJEŽBE.....	154
12.5.1	<i>Tijek izvođenja vježbe</i> .....	156
12.6	ZAVRŠNA PITANJA I ZADACI.....	157
12.6.1	<i>Završna pitanja u vezi izložene materije vježbe</i> .....	157
12.7	LITERATURA I DODATNI IZVORI.....	158
<b>VJEŽBA 13: UGNIJEŽDENI UPITI   ..... 159</b>		
13.1	MOTIVACIJA.....	159
13.2	CILJEVI VJEŽBE – ISHODI UČENJA.....	160
13.3	SAMOSTALNA PRIPREMA VJEŽBE.....	160
13.4	TEORIJSKI DIO PRIPREME VJEŽBE.....	161
13.4.1	<i>Ugniježdeni upiti - uvod</i> .....	161
13.4.1.1	Podupiti liste.....	162
13.4.1.2	Podupiti s predikatom usporedbe.....	162
13.4.1.3	Korelirani podupiti.....	162
13.4.1.4	Podupiti s predikatom postojanja (EXISTS, NOT EXISTS).....	163
13.5	SADRŽAJ I TIJEK IZVOĐENJA VJEŽBE.....	163
13.5.1	<i>Tijek izvođenja vježbe</i> .....	163
13.6	ZAVRŠNA PITANJA I ZADACI.....	169
13.6.1	<i>Završna pitanja</i> .....	169
13.7	LITERATURA I DODATNI MATERIJALI.....	170
<b>VJEŽBA 14: MYSQL DBMS, MYSQL WORKBENCH   ..... 171</b>		
14.1	MOTIVACIJA.....	171
14.2	CILJEVI VJEŽBE – ISHODI UČENJA.....	172
14.3	SAMOSTALNA PRIPREMA VJEŽBE.....	172
14.4	TEORIJSKI DIO PRIPREME VJEŽBE.....	173
14.4.1	<i>Uvod u MySQL SUBP</i> .....	173
14.4.2	<i>MySQL instalacija</i> .....	174
14.4.3	<i>MySQL Workbench</i> .....	185
14.4.4	<i>Uvoz podataka iz CSV datoteka</i> .....	195
14.5	SADRŽAJ I TIJEK IZVOĐENJA VJEŽBE.....	198
14.5.1	<i>Tijek izvođenja vježbe</i> .....	198
14.6	ZAVRŠNA PITANJA I ZADACI.....	202
14.6.1	<i>Zadaci za samostalno produblivanje znanja</i> .....	202
14.7	LITERATURA I DODATNI MATERIJALI.....	202



### III. Popis vježbi

VJ. 1:	UVOD U MICROSOFT ACCESS .....	15
VJ. 2:	KREIRANJE TABLICA U MS ACCESS-U .....	31
VJ. 3:	QUERY, POGLEDI, OGRANIČENJA .....	41
VJ. 4:	PROGRAMSKO SUČELJE BP – FORME .....	55
VJ. 5:	PROGRAMSKO SUČELJE BP – SUBFORME I IZVJEŠTAJI .....	67
VJ. 6:	SQL KOMANDE – DDL .....	81
VJ. 7:	ODBC .....	95
VJ. 8:	SQL DML .....	107
VJ. 9:	SQL SPOJEVI .....	121
VJ. 10:	FIREBIRD – INSTALACIJA I UPOTREBA .....	127
VJ. 11:	SQL OPERATORI I NULL VRIJEDNOST .....	137
VJ. 12:	SQL AGREGACIJE, SORTIRANJE .....	147
VJ. 13:	UGNIJEŽDENI UPITI .....	159
VJ. 14:	MYSQL DBMS, MYSQL WORKBENCH .....	171

## IV. Upute za korištenje priručnika

Vježbe izložene u ovom priručniku zahtijevaju od studenata da aktivno sudjeluju u samostalnoj pripremi, izvođenju vježbe u računalnom laboratoriju, te u naknadnom utvrđivanju ili dovršavanju vježbe. Planom kolegija baze podataka, predviđen je način bodovanja ukupne aktivnosti studenata, što će se pribrojiti konačnom uspjehu. Vježbe su strukturirane tako da svaka od njih sadrži niže navedene dijelove.

**Motivacija:** Obuhvaća kratko objašnjenje motivacije za sadržaj i izvođenje vježbe.

**Ciljevi vježbe – ishodi učenja:** Navedeno je što se očekuje da bi student putem vježbe trebao naučiti odnosno znati. Očekivane razine znanja: shvatiti/razumjeti, moći primijeniti, dobiti uvid/informaciju.

**Samostalna priprema vježbe:** Navedeno je koja su znanja potrebna, a koja nužna za pristupanje izvođenju vježbe u labu. Od studenata se očekuje da se samostalno pripreme za vježbu, i to prije nego što pristupe izvođenju vježbe u labu. Studenti bi trebali znati i razumjeti odgovore na postavljena pitanja. Osobito je bitno savladati znanja koja su navedena kao minimalno potrebna. Studenti će biti kroz blic-test ispitani kako bi se utvrdilo jesu li savladali priprema znanja. Ukoliko student nije u stanju razumjeti pripremu vježbe, trebao bi tražiti pomoć od nastavnika koji vodi lab, ali prije same vježbe .

**Teorijski dio vježbe:** U teorijskom dijelu vježbe je u skraćenoj formi navedena teorijska i druga materija, odnosno znanja i činjenice potrebne za izvođenje vježbe. Ideja je u tome da se studentima omogući jednostavnija i fokusirana priprema. Ovaj dio student treba dobro proraditi i naučiti prije dolaska u lab na izvođenje vježbe (u samostalnoj pripremi).

**Sadržaj vježbe:** Obuhvaća opis onoga što se na vježbi uči u praktičnom smislu.

**Tijek izvođenja vježbe:** Naveden je precizni tijek aktivnosti koje se izvode u labu.

**Završna pitanja:** Završna pitanja omogućuju studentu da provjeri u kolikoj mjeri je savladao vježbu. Ako student ne razumije pitanje ili ako ne zna odgovor na njega, onda je nešto propustio bilo u pripremi, bilo u izvođenju vježbe. Očekuje se da student sam ili timski s drugim studentima taj dio naknadno savlada nakon završetka vježbe u labu. Ova pitanja su osnova za znanje koje će se testirati za svaku prethodnu vježbu.

**Zadaci za produblivanje znanja:** Na kraju su navedena pitanja i zadaci za napredne studente koji žele svoja znanja produbiti.

**Literatura i dodatni izvori:** Ovaj dio sadrži popis osnovnih i dodatnih izvora za učenje fokusiranih na tekuću vježbu, ali i izvora koji omogućuju samostalno stjecanje dodatnih proširenih znanja.



## V. Literatura

Za izučavanje baza podataka postoji mnoštvo izvora. Kao temelj za teorijsku podlogu napravili smo sljedeći kratki izbor, koji po našem mišljenju pored referenci dodatnih izvora na kraju svake vježbe, daje dostatnu osnovu za razumijevanje baza podataka.

1. **Date, C. J. (2003). An Introduction to Database Systems, Eighth Edition, Addison-Wesley, 2003. ISBN 0-321-19784-4.**

*Ovo je kao što smo već naveli, jedan od najstarijih udžbenika i vjerojatno je izdan u najviše primjeraka kroz njegovih devet izdanja u proteklih više od trideset godina. Udžbenik nudi sveobuhvatan uvid u područje, lako je čitljiv i daje iscrpan pregled literature kroz povijest s komentarima.*

2. **Date, C. J. (2011). SQL and relational theory: how to write accurate SQL code. " O'Reilly Media, Inc."**

*Ova knjiga je nešto što po našem mišljenju niti jedan ozbiljan praktičar ne smije propustiti. Glavna tema je kako u uvjetima očitih i bitnih odstupanja SQL Standarda i SUBP u odnosu na teoriju relacijskog modela, ipak iskoristiti najviše što ti sustavi pružaju, a pri tome se držati relacijske teorije. Dakle, autor pokušava dati odgovore kako nesavršene alate koristiti maksimalno korektno u teorijskom smislu.*

3. **Radovan, M. (1993). Baza podataka-Relacijski pristup i SQL. Informator, Zagreb.**

*Radovanova knjiga je pisana kao solidan, kompletan udžbenik za uvod u baze podataka. Jasno je i teorijski korektno izložen relacijski model podataka kao „relacijski pristup“, te su izloženi osnovni koncepti SQL jezika. Materija je izložena jasno i na jednostavan način, ali ni u kojem slučaju naivno.*

4. **Tkalac, S. (1993). Relacijski model podataka. Društvo za razvoj informacijske pismenosti (DRIP) , Zagreb.**

*Ovo je udžbenik u kome je s teorijskog stajališta, najdublje obrađena materija teorije relacijskog modela napisana na hrvatskom jeziku. Preporučujemo ga za napredne studente, ali je pritom poželjna određena razina prethodnog znanja o bazama podataka i relacijskom modelu.*

5. **Varga, M. (1994). Baze podataka: konceptualno, logičko i fizičko modeliranje podataka. Društvo za razvoj informacijske pismenosti, Zagreb.**

*Ovaj je udžbenik lako pristupačan početnicima, koji se s bazama podataka nisu prije susretali. U tom smislu ga svakako preporučujemo.*



## Vježba 1: Uvod u Microsoft Access

### 1.1 Motivacija

Studenti koji se prvi puta susreću s teorijskim konceptima baza podataka, imaju poteškoća u smislu predodžbe, praktične realizacije, organizacije i načina upotrebe baze podataka. Teoriju baza podataka nije moguće primijeniti bez odgovarajućih alata, prije svega sustava za upravljanje bazom podataka (SUBP), koji se često konceptijski ali i sadržajno značajno razlikuju. Iako Microsoftov Access nije u teorijskom smislu, idealan primjer relacijskog SUBP, njegova jednostavnost, prihvatljiva relacijska vjernost te sveobuhvatnost kao alata za razvoj aplikacija zasnovanih na bazi podataka, po našem mišljenju ga izdvaja kao dobar alat za početno praktično učenje o bazama podataka. U ovoj vježbi želimo kroz naširoko poznati primjer aplikacije „Northwind Traders“ pokazati kako izgleda jednostavna aplikacija zasnovana na bazi podataka. Smatramo da studenti mogu kroz taj primjer steći predodžbu ne samo o strukturi i organizaciji baze podataka, nego i o odgovarajućoj aplikaciji koja obuhvaća korisničke forme i izvještaje te ostale objekte u bazi.

### Sadržaj vježbe:

<b>VJEŽBA 1:</b>	<b>UVOD U MICROSOFT ACCESS</b>	<b>15</b>
1.1	MOTIVACIJA	15
1.2	CILJEVI VJEŽBE – ISHODI UČENJA	16
1.3	SAMOSTALNA PRIPREMA VJEŽBE	16
1.4	TEORIJSKI DIO PRIPREME VJEŽBE	17
1.4.1	Microsoft Access – uvod .....	17
1.4.2	Razvoj Ms Access-a .....	18
1.4.3	Arhitektura Ms Accessa .....	18
1.4.4	Komponente Ms Accessa.....	20
1.5	SADRŽAJ I ZADATAK VJEŽBE	21
1.5.1	Izbornici i alatne trake.....	21
1.5.2	Aplikacija „Northwind Traders“.....	24
1.5.3	Ijeka izvođenja vježbe.....	27
1.6	ZAVRŠNA PITANJA I ZADACI	28
1.6.1	Završna pitanja u vezi izložene materije vježbe.....	28
1.6.2	Zadaci za dodatno samostalno produblivanje znanja .....	29
1.7	LITERATURA I DODATNI IZVORI	29

## 1.2 Ciljevi vježbe – ishodi učenja

Od studenata se očekuje da nakon uspješno završene vježbe:

- Shvate elemente/dijelove Ms Accessa kao aplikacijskog razvojnog alata.
- Znajju samostalno otvoriti postojeću Access aplikacijsku datoteku.
- Na temelju predloška (template), znaju generirati Access aplikacijsku datoteku.
- Razumiju svrhu i namjenu Ms Accessa.
- Razumiju i znaju koristiti osnovne postavke Ms Accessa.
- Dobiju grubi pojam o tome što je to baza podataka u praktičnom smislu.
- Razumiju logičku strukturu i sadržaj izloženog primjera baze podataka.

## 1.3 Samostalna priprema vježbe

Obaveza studenata je pripremiti se za aktivno sudjelovanje u izvođenju vježbe:

- Proučite prethodnu materiju izloženu na predavanjima iz kolegija BP.
- Dobro proučite teorijski dio ove vježbe.
- Potražite dodatne informacije na internetu kojima možete produbiti potrebno znanje.
- Osim teorijske podloge za pripremu ove vježbe, pokušajte shvatiti i ostali dio materijala.
- Ako su vam neki dijelovi materijala eventualno nerazumljivi pripremite pismena pitanja.
- Prije pristupanja vježbi, minimalno biste trebali znati odgovore sljedećih pitanja:
  - Što je baza podataka? Što je DBMS/SUBP ?
  - Kakva je sličnost/povezanost, a kakva je razlika među njima?
  - Što baza podataka sadrži?
  - Što je tip podatka? Koje osnovne tipove podataka poznajete?
  - Što je podatak, a što informacija, u čemu je sličnost i razlika?
  - Kako su podaci u bazi organizirani?
  - Što je Ms Access, čemu služi, od kojih komponenti se sastoji?

## 1.4 Teorijski dio pripreme vježbe

### 1.4.1 Microsoft Access – uvod

Microsoft Access, ili kako se novije inačice službeno zovu: Microsoft Office Access, integrirani je alat za razvoj aplikacija zasnovanih na osobnoj bazi podataka. Mnogi Access nazivaju bazom podataka. Međutim, Access je oduvijek bio više od baze podataka. Access predstavlja na određen način RAD tj. alat za brzu izradu aplikacija zasnovanih na bazi podataka. RAD je skraćenica od pojma „Rapid Applications Development“ (brzi razvoj aplikacija). Arhitektura i način upotrebe RAD alata omogućuje da se u toku razvoja aplikacija razvijatelj (developer/projektant) primarno fokusira na brzinu razvoja, čak i na uštrb performansi aplikacije i/ili na uštrb funkcionalnosti.

RAD alati u pravilu, omogućuju jednu od dvije metode prototipiziranja. Prvi tip se naziva „razvij pa odbaci“. Drugi tip se naziva „razvij pa poboljšavaj“. U prvom slučaju prototip će poslužiti kako bi se na efikasan način dobila dobra funkcionalna specifikacija – tj., kako bi se spoznalo kakva bi u funkcionalnom smislu aplikacija trebala biti. Nakon izrade ovakvog prototipa može se pristupiti izradi potrebne aplikacije, budući da takav prototip ne daje dovoljno dobre performanse u stvarnoj upotrebi, a prototip se odbacuje. Ovakav model razvoja prototipiziranjem se koristi u situacijama kada je teško ili nije moguće ekonomično utvrditi specifikaciju funkcionalnih zahtjeva za potrebnu aplikaciju. On je također pogodan pristup u situacijama kad naručitelji aplikacije odnosno njeni korisnici, ali također i projektanti nemaju jasnu predodžbu o tome kako bi aplikacija trebala funkcionirati. Prototipiziranjem se u takvim slučajevima postupno dobiva pouzdana specifikacija zahtjeva uz minimalni napor naručitelja, korisnika i projektanta. Na taj se način značajno smanjuje rizik neuspjeha konačne aplikacije, koja se gradi na temelju dobivenog prototipa.

Drugi oblik prototipa omogućuje da se dobije aplikacija koja će dati u početku slabije, ali ipak minimalno prihvatljive performanse, uz uvjet da se kroz daljnje poboljšavanje može dobiti potpuno prihvatljivo rješenje. Access je pogodan za oba slučaja prototipiziranja, i to zbog toga što sadrži sve elemente potrebne za razvoj, integrirane u konzistentan alat. Kod tako dobivenih rješenja će najčešće biti kritična točka u pogledu performansi sama baza podataka. Međutim, Access omogućuje jednostavnu, gotovo automatsku nadogradnju („upsizing“) na SQL Server ili pak putem ODBC sučelja moguće je povezivanje na praktično bilo koji visokoperformansni DBMS.

Potrebno je napomenuti da je Ms Access osim što omogućuje profesionalni pristup razvoju aplikacija, omiljen alat koji koriste i napredniji korisnici za razvoj manjih lokalnih aplikacija. Jasno, u ovom drugom slučaju aplikacije će najvjerojatnije biti neprihvatljive prema strogim profesionalnim kriterijima, ali ipak upotrebljive i korisne. Treba napomenuti da se Microsoft u svojim materijalima u prilogu za korištenje Accessa najčešće usredotočuje na ovu drugu skupinu korisnika. Upravo zbog toga, Access je na lošem glasu kod nekih profesionalaca koji ga nedovoljno poznaju. Unatoč tome on predstavlja jedinstven RAD

alat na tržištu . Činjenica je da mu niti jedan sličan RAD alat u proteklih dvadeset godina nije preuzeo primat u toj tržišnoj niši.

Ove osobine jednostavnosti korištenja za početnike, također ga svrstavaju u često korišten alat za prvi susret sa bazama podataka, odnosno za početno učenje baza podataka i SQL jezika. Pritom treba imati na umu da Ms Access nije DBMS, nego jedan cjeloviti razvojni alat koji nikako ne treba uspoređivati s drugim robusnijim i kompleksnijim, ali i neusporedivo skupljim sustavima za upravljanje bazom podataka poput SQL Servera, Oracle-a DB2 itd.

### 1.4.2 Razvoj Ms Access-a

Prva inačica (verzija) Accessa pojavila se 1992. godine. Temelj na kome je on razvijen bio je vrlo ozbiljan. Microsoft je 80-tih godina pokrenuo projekt „Omega“ čiji je cilj bio razvoj tržišno konkurentnog relacijskog sustava za upravljanje bazama podataka (RDBMS). Budući da je Microsoft kasnije od tog sustava odustao, dobivena tehnologija bila je upotrijebljena za razvoj Accessa. U to vrijeme Microsoft je favorizirao BASIC programski jezik „kao programski jezik za mase“ pa je logična posljedica bila da se on uključi u Access projekt. Pojava Accessa nije imala dovoljno pogodno tlo, budući da Microsoft nije imao dovoljno dobar 32-bitni operativni sustav. Pojava Windowsa 95 i nakon toga Microsoft Office 95 paketa čiji je Access bio dio, značilo je prekretnicu u tržišnom nastupu Accessa.

Sastavni dio paketa i glavni programski jezik Accessa je bio Visual Basic for Applications (VBA). VBA je je prošireni podskup Visual Basica, koji je Accessu omogućio stvaranje procedura koje inače nisu bile moguće u jednostavnijoj verziji SQL-a koje Access podržava. Nakon Verzije 95, pojavila se inačica 97. Međutim, sljedeću prekretnicu je predstavljala inačica Access 2000. Uslijedile su inačice Access XP i Access 2003. Ove inačice (verzije) su bile zasnovane na Jet DBMS-u (Jet Engine).

Od inačice 2007, preko 2010 pa do zadnje inačice 2013, Jet DBMS je napušten, a njegova arhitektura je promijenjena. Promijenjena je i razvojna okolina u skladu s Office konceptom. Posljednja inačica je usredotočena na web aplikacije korištenjem Microsoftovog Sharepoint servisa a u osnovi bi SQLServer trebao biti DBMS . Njena desktop verzija je u velikoj mjeri kompatibilna s inačicom 2007 odnosno 2010. Za ove vježbe smo odabrali 32-bitnu inačicu Access 2010, prije svega zbog kompatibilnosti sa prethodnom verzijom OS Windows xp, kako bi bila dostupnija studentima. Inačice Accessa počevši od 2007, mogu podržavati 32 ili 64 bitno adresiranje. 64 bitna verzija zahtijeva 64 bitni Windows operativni sustav, dok 32 bitna verzija može raditi i pod 32 bitnim, ali pod 64 bitnim Windowsima.

### 1.4.3 Arhitektura Ms Accessa

Kao što je poznato, s obzirom na namjenu, danas postoje dvije osnovne arhitekture odnosno izvedbe baza podataka, odnosno SUBP. Jedno su baze podataka koje su utemeljene na softverskom

sustavu koji omogućuje cjelovito i konzistentno upravljanje bazom podataka i njenim sadržajem u vidu softvera-servera (DBMS). Ova vrsta omogućuje konzistenciju, cjelovitost i skalabilnu djeljivost podataka u bazi, i pritom omogućuje posluživanje većeg broja korisnika odnosno klijentskih programa. Druga vrsta DBMS-a nije zasnovana na samostalnom serveru, već su funkcije DBMS-a ugrađene u svaku aplikaciju. Ova vrsta baza podataka se naziva personalna odnosno desktop baza. Namijenjena za aplikacije koje opslužuju jednog korisnika na lokalnom računalu ili eventualno manji broj konkurentnih korisnika. Ms Access pripada u tu drugu kategoriju, dok SQLServer u prvu kategoriju RDBMS-a (relacijskih sustava za upravljanje bazom podataka). Pojam baza podataka se često koristi i kao sinonim za podatke kojima upravlja DBMS, ali se često i koristi kao sinonim za sam DBMS, odnosno DBMS i podaci.

Access aplikacija se sprema u jednu datoteku koja sadrži sve objekte aplikacije (forme, izvještaje, programske module,...) kao i bazu podataka i njene objekte te njihov sadržaj (podatke). Takva datoteka standardno ima ekstenziju **accdb**. U prethodnim verzijama (prije Accessa 2007) ta je ekstenzija bila **mdb** i imala je nešto drugačiju strukturu od **accdb**. Inačica Access 2010 podržava oba formata. Pored izvedbe aplikacije u kojoj se aplikativni softver nalazi zajedno s objektima baze podataka (tj. podacima) u istoj datoteci, Access omogućuje da se izvršna aplikacija razdvoji od podataka u dvije odvojene datoteke. U pravilu je gotovo uvijek, dobro razdvojiti aplikacijske objekte u zasebnu datoteku, a bazu podataka u drugu datoteku. Datoteku koja sadrži aplikacijske objekte treba linkati (povezati) s datotekom koja sadrži bazu podataka. Obje datoteke mogu imati istu **accdb** ekstenziju. Na taj način se osigurava bolja upravljivost, ali pouzdanost u višekorisničkom radu.

Aplikacijska datoteka je otvorena za korisnike tako da oni mogu neposredno mijenjati aplikacijske i ostale objekte. Ovo može biti dobro, ali često razvijatelj aplikacije želi osigurati da korisnici mogu samo koristiti bazu, te da pritom ne mogu mijenjati sadržaj aplikacije. Access omogućuje izradu zaštićene verziju izvršne datoteke koja za korisnike djeluje kao da se radi o izvršnom programu koji korisnici ne mogu mijenjati. To se može postići tzv. publiciranjem (tj., spremanjem u drugi format: Save & Publish) Access aplikacije kao kompajlirane i zaštićenu datoteku s ekstenzijom „**accde**“.

Kompajlirana i zapakirana verzija ACCDE ima još jednu prednost. Moguće ju je izvoditi na računalu na kojem nije instalirana potpuna verzija Accessa. Umjesto toga potrebno je instalirati tzv. run-time verziju Accessa koja je besplatna i može se skinuti s Microsoftove web-stranice. Na taj način se postiže dodatna sigurnost, stabilnost i jednostavnost distribucije Access aplikacija. Ova mogućnost u punoj formi, postoji počevši od inačice Access2007.

Također treba napomenuti da datoteke s ekstenzijom **mdb** ili **accdb**, mogu poslužiti kao eksterni izvori podataka odnosno baze podataka, za druge programe. Tako je na primjer, iz Excela moguće direktno, bez programiranja, čitati i upisivati podatke u te datoteke kao eksterne baze podataka.

### 1.4.4 Komponente Ms Accessa

Access se sastoji od sljedećih dijelova:

- DBMS-engine,
- Query designer,
- Form-designer,
- Report – designer,
- Visual Basic IDE sa kompajlerom,
- Macro editor.

**DBMS engine** je relacijski dbms. To znači da je njegov inherentni model podataka relacijski model podataka. On osigurava konzistenciju i integritet podataka, ali i jednostavnost primjene. DBMS dio Accessa je relacijski, što u osnovi znači da:

- Podatke sprema u tabličnu strukturu – tablice
- Za manipulaciju podacima (ubacivanje, mijenjanje, brisanje i dohvaćanje) koristi SQL jezik
- Operacije/manipulacije podacima se izvode nad tablicama.

**Query Designer** je dio Accessa koji omogućuje dvojni način kreiranja i mijenjanja SQL upita:

- Neposrednim pisanjem SQL modula,
- Korištenjem tabelarnog alata koji se obično naziva QBE (Query by Example), koji na vrlo jednostavan način omogućuje stvaranje čak i složenih SQL upita.

**Form designer** je alat koji omogućuje vizualno kreiranje programskih formi koje predstavljaju dinamičko sučelje za pristup podacima u bazi (ubacivanje, mijenjanje, brisanje i prikaz).

**Report Designer** je alat koji na vizualan način omogućuje stvaranje izvještaja (prikaza) podataka iz baze podataka. On ima funkciju pretpregled, koja omogućuje prikaz izvještaja na ekranu ali i funkciju ispisa na pisač ili transfera u PDF i druge formate.

**Visul Basic IDE** je interaktivni razvojni editor za programiranje u Accessovoj verziji Visula Basic jezika. Ovaj jezik je pravi programski jezik s mnogim naprednim osobinama, ali i proširenjima potrebnim za rad s Access bazom. Zapravo se radi o tzv. Visual Basic for Applications jeziku (VBA). Koristi se za programiranje rutina koje se mogu pridružiti formama i izvještajima. On također omogućuje korištenje ugrađenog dinamičkog SQL koda. VBA je standardni ugrađeni jezik za sve Microsoft Office aplikacije.

**Makro editor** je editor takozvanih makro-programa. Radi se o alatu koji na intuitivan način omogućuje razvoj programskih rutina za korisnike koji nisu programeri i ne poznaju Visual Basic. Performanse ovakvih programa su obično lošije od onih izrađenih u VBA, ali postoji mogućnost automatskog pretvaranja makro programa u VBA-program pomoću ugrađenog alata.

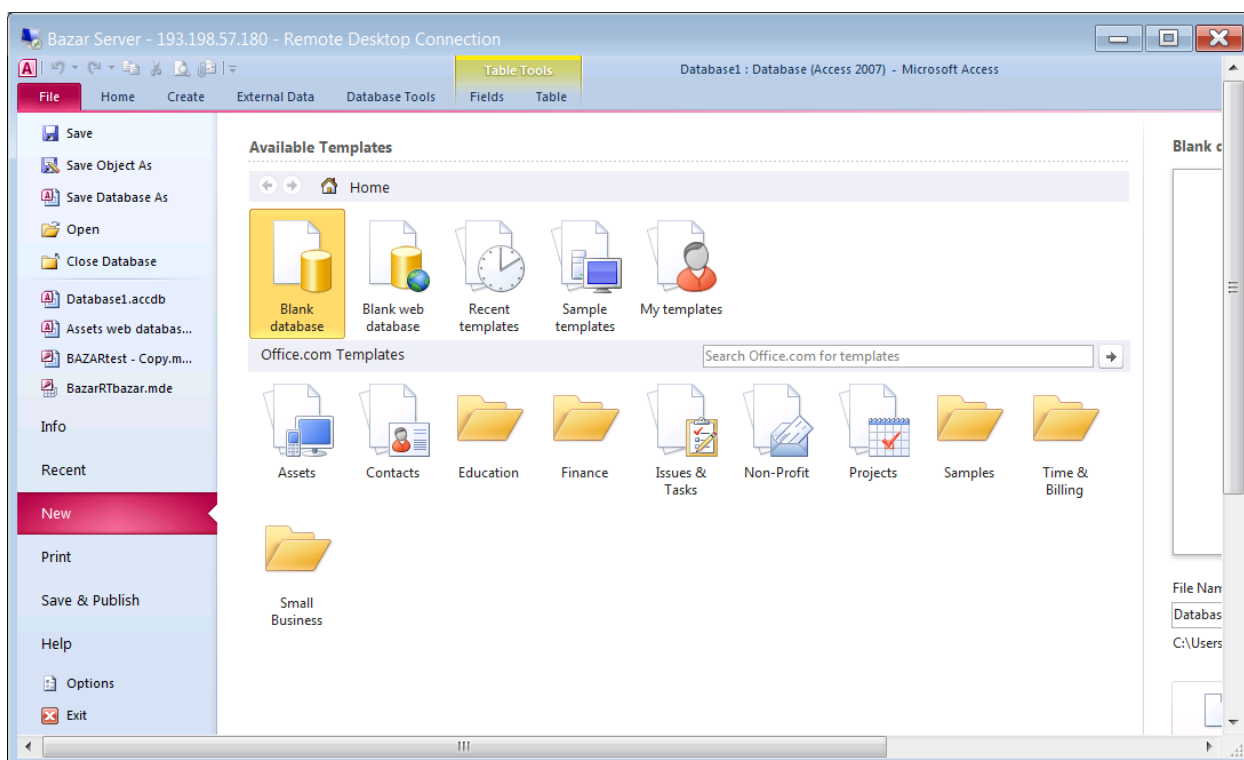


## 1.5 Sadržaj i zadatak vježbe

### 1.5.1 Izbornici i alatne trake

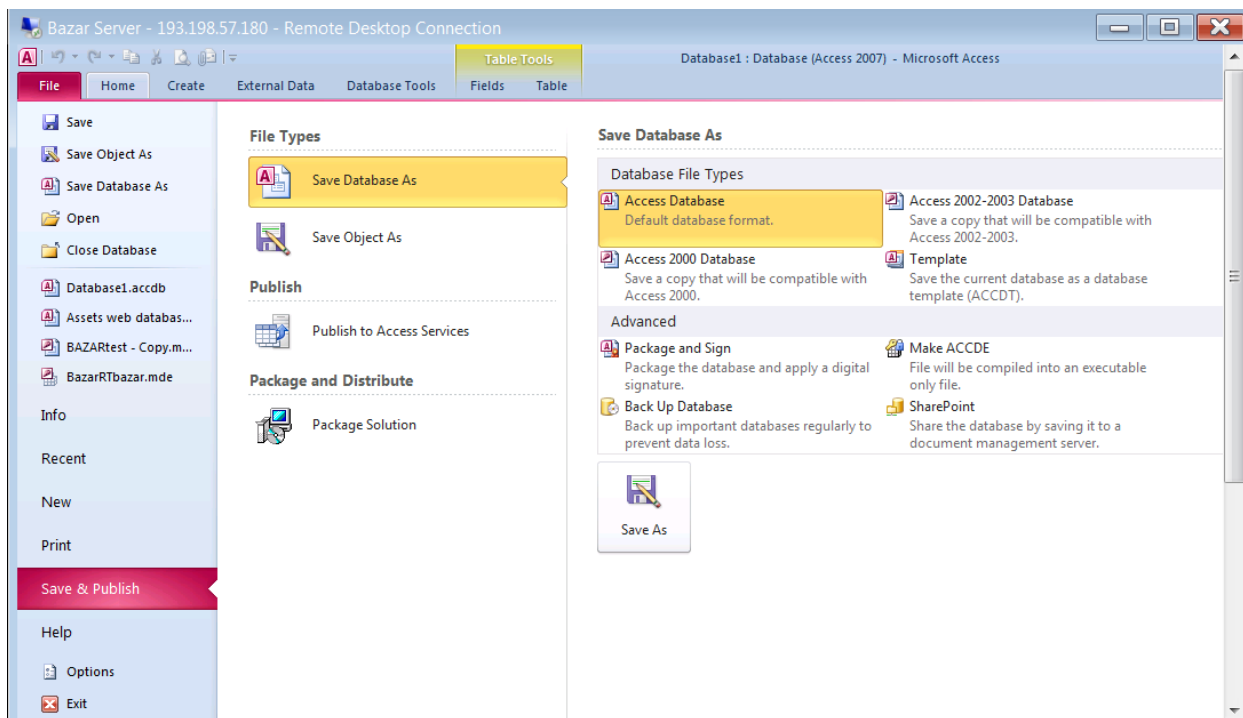
Razvoj i korištenje neke aplikacije u Accessu zahtijeva da se aplikacija najprije kreira, a kod svake sljedeće upotrebe ponovno otvori. Kreiranje nove aplikacije pretpostavlja stvaranje/otvaranje nove Access datoteke te nakon toga kreiranje objekata koje će aplikacija sadržavati. Access nudi cijeli niz gotovih predložaka koji sadrže inicijalnu definiciju kako podatkovnih tako i aplikacijskih objekata, odnosno gotovih aplikacija za pojedina područja poslovanja. To omogućuje da se takvi predlošci (templates) učitaju te generiraju polaznu aplikaciju koja se nakon toga može prilagoditi specifičnim potrebama. Kao što smo u teorijskom dijelu naveli, postoje dvije grupe objekata: objekti baze podataka i aplikacijski objekti. U Accessu najprije kreiramo objekte baze podataka: relacijske tablice, SQL upite i poglede, te integritetska ograničenja. Baza podataka predstavlja logički podatkovni model poslovnog sustava, odnosno onoga dijela tog sustava za koji se baza gradi.

Prikaz izbornika **File->New**, koji omogućuje kreiranje nove baze na osnovu predložaka:



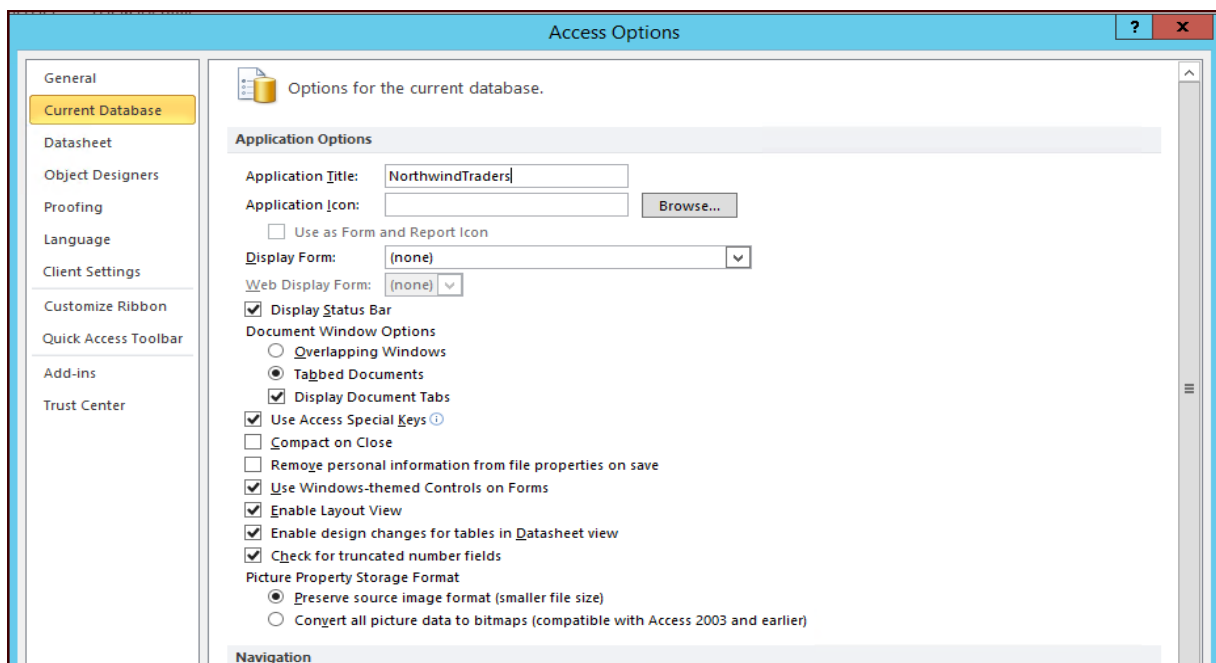
Access aplikaciju možemo spremiti u formi u kojoj je kreirana, ili u više drugih oblika koji su ponuđeni u izborniku. Pored spremanja u formate prethodnih verzija, osobito je značajno spremanje u kompajliranu verziju tipa **accde**, koja kao što smo prije rekli, postaje izvršni kompajlirani i slobodno prenosivi kod. Tu je također mogućnost digitalnog potpisivanja aplikacije te isto tako i mogućnost dobivanja web verzije koja se izvršava uz pomoć SharePoint servisa.

Prikaz izbornika Save & publish (spremi i publiciraj) s mogućim opcijama izlaznog formata:



Access omogućuje cijeli niz postavki parametara koji imaju utjecaj na njegovo funkcioniranje kako u toku razvoja aplikacije, tako i na konačni izgled i način funkcioniranja korisničke aplikacije. Postavke se definiraju u izborniku: Access Options, a organizirane su u jedanaest grupa.

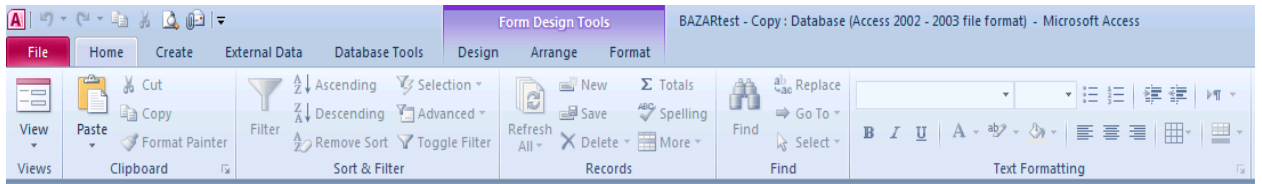
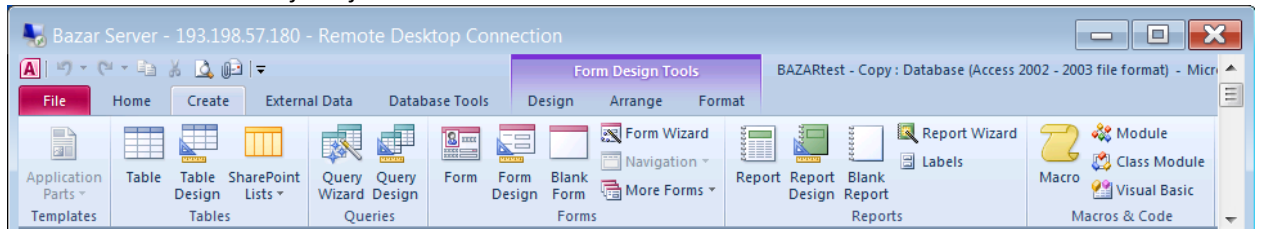
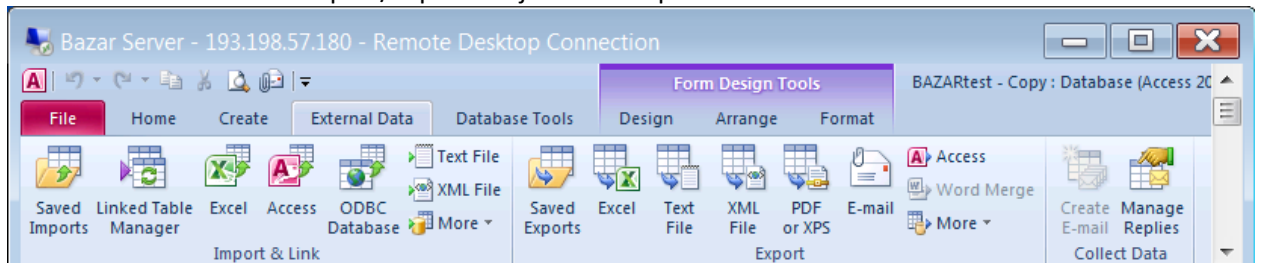
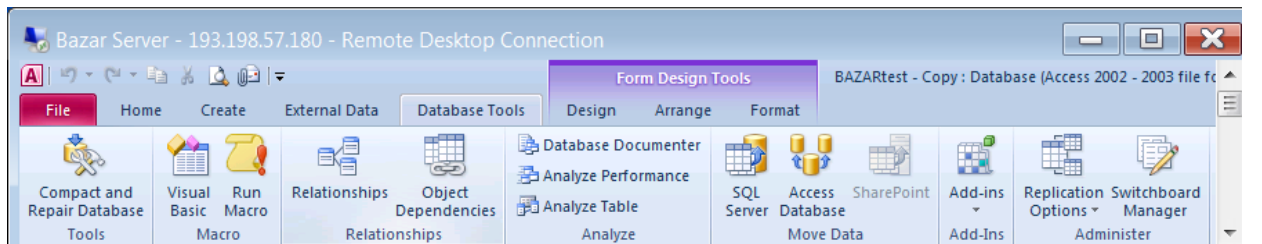
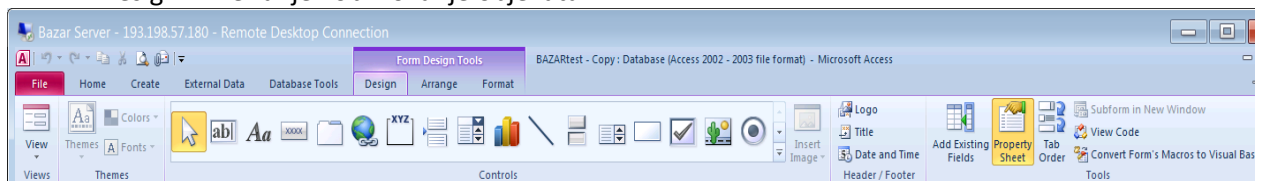
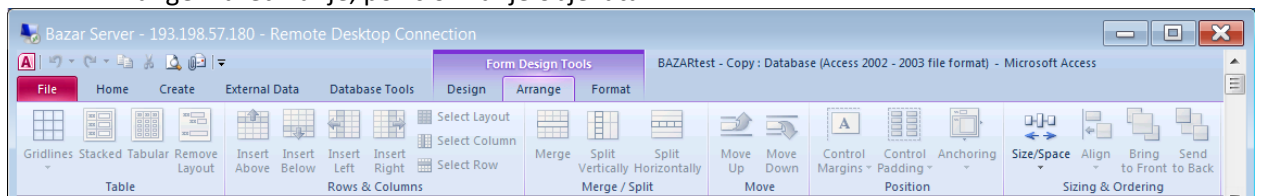
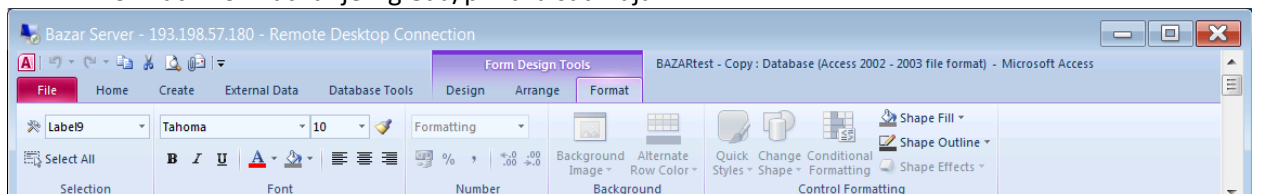
Prikaz izbornika s grupama opcija za podešavanje parametara rada Accessa:



Tijekom razvoja i korištenja Access aplikacije na raspolaganju je velik broj alata (tools) dostupnih preko ikona na odgovarajućoj alatnoj traci (ribbon). Postoji više alatnih traka s funkcionalno grupiranim ikonama alata, koje su kontekstno osjetljive. To znači da će alati biti aktivni samo ako to ima smisla s obzirom na trenutni kontekst aplikacije.

**Prikaz alatnih traka:**

Home:

**Create – kreiranje objekata****Eksternal Data – import/export vanjskih izvora podataka:****Database Tools – alati za rad s bazom:****Design – kreiranje i oblikovanje objekata:****Arrange – uređivanje, pozicioniranje objekata:****Format – formatiranje izgleda/prikaza sadržaja:**

## 1.5.2 Aplikacija „Northwind Traders“

U primjeru baze podataka koja će se koristiti za ovu vježbu, u Accessu je izrađena aplikacija zasnovana na bazi podataka, koja predstavlja pojednostavljeni model poslovanja fiktivne tvrtke „**Northwind Traders**“ (u nastavku NWT). Već od prvih verzija Accessa ova se aplikacija koristi kao ilustrativni cjeloviti primjer aplikacije u Accessu. Značaj primjera je u tome što na jednostavan način ilustrira aplikaciju za podršku poslovanja u fiktivnom/zamišljenom poduzeću NWT.

Primjer je razumljiv, previše pojednostavljen da bi se stvarno mogao koristiti u živom poduzeću. Međutim, on može poslužiti kao aplikacijski i podatkovni kostur za izgradnju stvarno upotrebljive aplikacije odnosno baze podataka. Nadalje, baza podataka na kojoj se zasniva, nije u smislu relacijske teorije potpuno korektna, ali je s obzirom na namjenu prihvatljiva.

Prikaz NWT bez navigacijskog okna (minimiziran je):

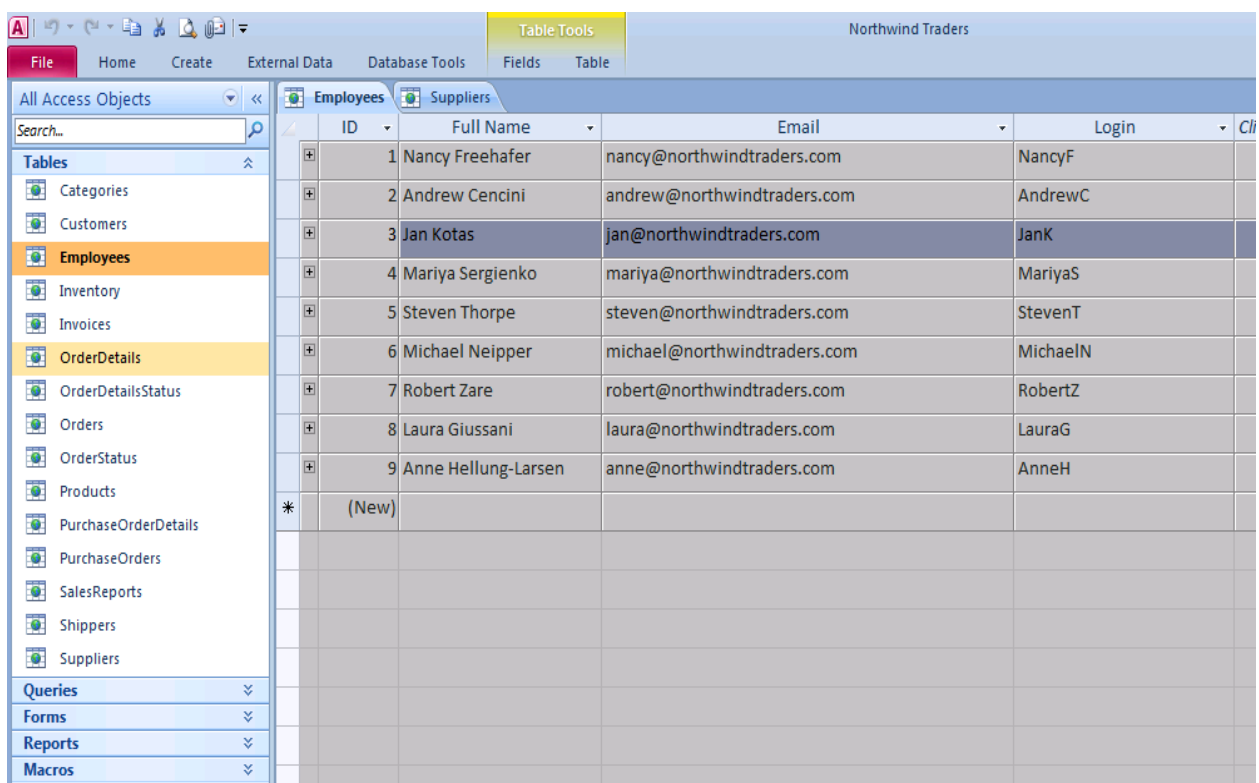
ID	Email	Full Name	Login
1	nancy@northwindtraders.com	Nancy Freehafer	NancyF
2	andrew@northwindtraders.com	Andrew Cencini	AndrewC
3	jan@northwindtraders.com	Jan Kotas	JanK
4	mariya@northwindtraders.com	Mariya Sergienko	MariyaS
5	steven@northwindtraders.com	Steven Thorpe	StevenT
6	michael@northwindtraders.com	Michael Neipper	MichaelN
7	robert@northwindtraders.com	Robert Zare	RobertZ
8	laura@northwindtraders.com	Laura Giussani	LauraG
9	anne@northwindtraders.com	Anne Hellung-Larsen	AnneH
* (New)			

Poslovni model NWT uključuje:

- NWT kao tvrtku koja ima zaposlenike (Employees), te robu (Products) koju čuva na zalih (Inventory)
- NWT nabavlja robu putem narudžbi (Purchase Orders)
- NWT prodaje robu kupcima (Customers) na temelju prodajnih narudžbi (Sales Orders) isporučujući robu putem transportera (Shippers), te za to ispostavlja račune (Invoices)
- O ostvarenoj prodaji se mogu ispisati prodajni izvještaji (Sales reports).

Kad se otvori Access aplikacija s ciljem daljnjeg razvoja odnosno promjene, ispod izbornika i alatne trake, prikazat će se dva prozora: prvi prozor je na lijevoj strani ekrana - to je navigacijski prozor. U njemu su prikazani podatkovni i aplikacijski objekti grupirani u grupe (elemente). Kad se pojedina grupa rastvori, prikazat će se objekti te grupe. Drugi prozor (desni) će prikazati onaj objekt koji je selektiran u navigacijskom prozoru. Istovremeno je moguć prikaz više objekata. Način prikaza se može odabrati putem izbornika koji se pojavljuje nakon što pritisnemo desnu tipku miša iznad naslova odnosno naziva odabranog objekta.

Na primjeru NWT popis tablica je prikazan kao lista s desne strane, a na lijevoj strani je prikaz sadržaja odabrane tablice (Employees):

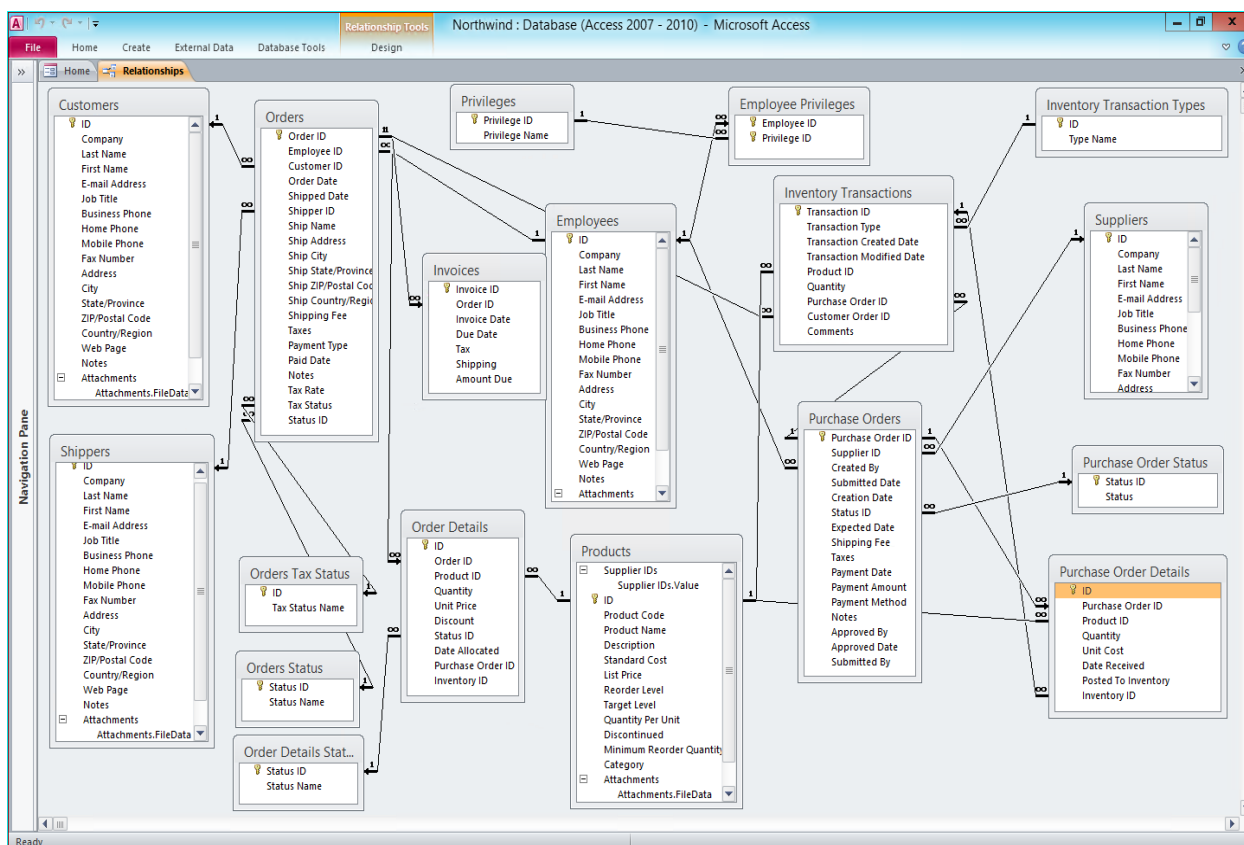


ID	Full Name	Email	Login
1	Nancy Freehafer	nancy@northwindtraders.com	NancyF
2	Andrew Cencini	andrew@northwindtraders.com	AndrewC
3	Jan Kotas	jan@northwindtraders.com	JanK
4	Mariya Sergienko	mariya@northwindtraders.com	MariyaS
5	Steven Thorpe	steven@northwindtraders.com	StevenT
6	Michael Neipper	michael@northwindtraders.com	MichaelN
7	Robert Zare	robert@northwindtraders.com	RobertZ
8	Laura Giussani	laura@northwindtraders.com	LauraG
9	Anne Hellung-Larsen	anne@northwindtraders.com	AnneH
*	(New)		

Temeljni dio svake Access aplikacije je baza podataka. Glavni objekt baze podataka su relacije ili relacijske tablice. Relacije sadrže entorke vrijednosti atributa karakterističnih za svaku relaciju. Te entorke su skupovi vrijednosti, koje se u relacijskoj tablici prikazuju kao redovi, dok se stupci odnose na homogene vrijednosti atributa. Relacije ili relacijske tablice ili jednostavno tablice, su međusobno logički povezane preko stranih i primarnih ključeva i odgovarajućih integritetskih pravila, o čemu će biti više riječi u idućim vježbama. Za sada napomenimo samo, da u relacijskoj bazi podaci odnosno relacije, nisu povezane fizičkim vezama, već su te veze isključivo logičke. Fizička i logička razina relacijske baze podataka su jasno razdvojene. Projektant/razvijatelj se bavi konceptualnom i logičkom razinom, a administrator baze podataka fizičkom razinom.

U NWT aplikaciji je implementirano 18 relacijskih tablica:

- Customers (Kupci) – označava kupce kojima se prodaje roba.
- Shippers (Transporteri) – koji isporučuju/transportiraju naručenu robu kupcima.
- Orders (Narudžbe) narudžbe koje ispostavljaju kupci –narudžba sadrži jednu ili više stavaka
- Orders Details (Stavke narudžbi) – svaka se stavka odnosi na naručeni artikl/robu i narudžbu
- Orders Detail Status (Status stavke narudžbe)
- Order Status (Status narudžbi)
- Order Tax Status (Porezni statusi narudžbi)
- Products (Roba) – proizvodi/roba koja se naručuje od dobavljača i prodaje kupcima
- Invoices (Računi) – računi kupcima za naručenu/prodanu/isporučenu robu
- Employees (Zaposlenici) – zaposlenici koji obrađuju narudžbe kupaca
- Pvilesges (Ovlasti) – prava/uloge koja zaposlenici mogu imati u aplikaciji
- Employees Privileges (Ovlasti zaposlenika)- prava karakteristična za pojedinog zaposlenika
- Purchase Orders (Narudžbe od dobavljača)
- Purchase Orders Details (Stavke narudžbi od dobavljača) –stavke narudžbi od dobavljača
- Purchase Orders Status (Statusi narudžbi od dobavljača)
- Suppliers (Dobavljači) – dobavljači od kojih se kupuje roba
- Inventory Transactions (Transakcije skladišnog prometa) – skladišni ulaz/izlaz robe
- Inventory Transactions Types (Tipovi transakcija skladišnog prometa)



### 1.5.3 Tijek izvođenja vježbe

1. Otvorite Access.
2. Učitajte aplikaciju „Northwind Traders“
3. Otvorite aplikaciju i proučite njen sadržaj: forme, izbornike, izvještaje, tablice
4. Uočite kako se aplikacija ponaša, u početku nemojte mijenjati podatke, samo ih analizirajte na logičkoj razini
5. Promijenite, izbrišite, dodajte podatke u prikazane forme – nemojte to činiti nasumce, već sa smislom u odnosu na poslovni model fiktivnog poduzeća “NWT”.
6. Uočite kako su podaci međusobno povezani
7. Što su podaci u ovoj aplikaciji, a što su informacije?
8. Prikažite izvještaje na ekranu, otisnite ih u pdf formatu datoteke.
9. Podesite opsijske parametre prema svojim preferencijama – proučite upute o značenju pojedinog parametra.
10. Prikažite aplikaciju pod novom opsijskim parametrima – ima li promjene?
11. Razdvojite aplikaciju na dvije datoteke (Database tools – split database)
  - a. NWTaplikacija.accdb,
  - b. NWTDb.accdb
12. Publicirajte NWTaplikacija.accdb kao NWTrt.accdb (Save/Publish – Make ACCDE)
13. Izvedite NWTrt.accdb
14. Koje ste razlike uočili?

**Napomena:** Prije nego što počnete dodavati ili mijenjati podatke, spremite aplikaciju pod nekim drugim imenom, kako ne biste izgubili izvorne podatke.

## 1.6 Završna pitanja i zadaci

Nakon aktivnog sudjelovanja i izvođenju vježbe u labu, potrebno je provjeriti je li vježba shvaćena u potpunosti. To ćete znati ako ste po završetku vježbe, u stanju samostalno i ispravno odgovoriti na završna pitanja (test u idućoj vježbi će se temeljiti na ovim pitanjima). Pored toga, poželjno je da samostalno pokušate produbiti i proširiti svoje znanje proučavajući navedene dodatne izvore. Također, pokušajte sami pronaći druge referentne izvore i materijale.

### 1.6.1 Završna pitanja u vezi izložene materije vježbe

- Što je to Microsoft Office Access?
- Kada je nastala njegova 1. inačica
- Koja je trenutno najnovija izdana inačica Ms Accessa?
- Koju inačicu/verziju koristite na vježbi, a koju kod kuće?
- Nabrojite funkcionalne dijelove Ms Accessa.
- Kolika je bitnost Accessa 2010?
- U kojoj okolini funkcionira MsAccess- uz koji OS?
- Funkcionira li MsAccess izvan Office sustava ili samo u njegovom sastavu?
- Koji model podataka podržava DBMS u Accessu?
- Što je temeljni objekt strukture u Ms Access-ovoj bazi podataka?
- Koji model podataka Access podržava? Je li on karakterističan samo za Access?
- Koji univerzalni jezik podataka se koristi u Accessu? Je li on karakterističan samo za Access?
- Kakav tip aplikacija je moguće razviti pomoću MsAccessa?
- Je li za taj razvoj potrebno posjedovati legalnu licencu – koju?
- Je li za njihovu primjenu potrebna licenca MsAccessa?
- Jesu li te aplikacije pogodne za intenzivan višekorisničku pristup - objasnite?
- Uključuje li/sadrži Access personalni/desktop ili serverski DBMS?
- Može li MS Access koristiti i druge baze podataka – objasnite?
- Čemu služe Forme u MsAccessu?
- Čemu služe izvještaji/reporti u Ms Accessu?
- Na kojem programskom jeziku je zasnovan MsAccess?
- Omogućuje li Access dijeljenje resursa baze podataka putem weba - kako?
- Što znači pojam RAD alat? Je li Access RAD alat?
- Što se postiže razdvajanjem Access aplikacije od baze podataka koju ona koristi?
- Koliko relacija sadrži aplikacija NWT?
- Što pojedine relacije predstavljaju u poslovnom modelu NWT?



## 1.6.2 Zadaci za dodatno samostalno produblјivanje znanja

- Pokušajte shvatiti koje atribute sadrže relacijske tablice (relacije) u NWT te kakvo je njihovo značenje.
- Pogledajte i proučite i neke druge aplikacije koje je moguće učitati kao predložak (Template).

## 1.7 Literatura i dodatni izvori

- 1) <http://office.microsoft.com/hr-hr/access-help/prijelaz-na-access-2010-RZ101791922.aspx?CTT=1>
- 2) <https://support.office.com/en-za/article/Ways-to-share-an-Access-database-2c24eb08-bee1-453e-be8e-455f847c5c74>
- 3) <https://www.youtube.com/watch?v=hMzf6u8hWqk>
- 4) <https://blogs.office.com/2010/07/19/northwind-2010-web-database-is-now-available/>



## Vježba 2: Kreiranje tablica u MsAccessu

### 2.1 Motivacija

U prethodnoj smo vježbi trebali naučiti kako generirati Access aplikaciju na temelju predloška. Vidjeli smo relacijsku/tabelarnu organizaciju baze podataka, ali bez ulaženja u detalje njene strukturalne komponente. U ovoj ćemo vježbi pokušati kreirati novu bazu podataka neposredno, bez predloška. Ideja je u tome da studenti na postupan način, nauče i shvate što je to strukturalna komponenta, te što su ograničenja na razini atributa u relacijskoj bazi podataka. Access je za to pogodan, jer se strukturalna komponenta može definirati putem jednostavnih alata u tabelarnoj formi, bez potrebe da se poznaje SQL jezik. Kasnije u idućim vježbama, definirat ćemo strukturu i ograničenja baze podataka putem SQL jezika. Ovom vježbom želimo postići da studenti kreiraju strukturu (relacijske tablice, attribute i primarne ključeve), te da na jednostavnim primjerima vide kako tako definirana struktura djeluje.

### Sadržaj vježbe:

#### KREIRANJE TABLICA U MSACCESS-U | 31

2.1	MOTIVACIJA	31
2.3	CILJEVI VJEŽBE – ISHODI UČENJA	32
2.4	SAMOSTALNA PRIPREMA VJEŽBE	32
2.5	TEORIJSKI DIO PRIPREME VJEŽBE	33
2.5.1	Tablice u Ms Access-u.....	33
2.5.2	Tipovi podataka.....	34
2.5.3	Ključevi relacije.....	35
2.5.4	Ograničenja i formati na razini atributa.....	37
2.6	SADRŽAJ I TIJEK IZVOĐENJA VJEŽBE	38
2.6.1	Sadržaj vježbe.....	38
2.6.2	Tijek izvođenja vježbe.....	39
2.7	ZAVRŠNA PITANJA I ZADACI	39
2.7.1	Završna pitanja u vezi izložene materije vježbe.....	39
2.7.2	Zadaci za samostalno produbljivanje znanja.....	40
2.8	LITERATURA I DODATNI MATERIJALI	40

## 2.2 Ciljevi vježbe – ishodi učenja

Od studenata se očekuje da, nakon uspješno završene vježbe:

- Razumiju pojam i svrhu relacijske tablice
- Shvate pojam strukture relacijske tablice
- Razumiju što su tipovi i domene podataka
- Znaju kreirati tablice u Accessu
- Znaju pravilno primjenjivati tipove podataka
- Znaju što je kandidatni ključ te kako se on određuje
- Razumiju što je primarni ključ relacije
- Znaju kako odabrati i definirati primarne ključeve u Accessu
- Razlikuju prirodne od surogatnih ključeva
- Razumiju što su to ograničenja na razini atributa

## 2.3 Samostalna priprema vježbe

Prije dolaska u lab, obaveze studenata su:

- Proučite prethodnu materiju sa predavanja iz kolegija BP
- Proučite pripremne materijale za ovu vježbu
- Utvrdite prethodnu vježbu – pitanja u prethodnoj vježbi će predstavljati temelj za blic-test u ovoj vježbi
- Minimalna znanja potrebna za pristupanje vježbi su:
  - Razumjeti što je to relacija, od kojih dijelova se sastoji
  - Razumjeti što je atribut, tip podatka, domena
  - Znati koji osnovni tipovi podataka postoje u Accessu
  - Razumjeti pojam i značaj ključeva relacije

## 2.4 Teorijski dio pripreme vježbe

### 2.4.1 Tablice u Ms Access-u

Glavni element strukture u Accessu su relacijske tablice. Access u velikoj mjeri podržava relacijski model podataka. Relacijske tablice su specijalan oblik tablica, tj., to su tablice koje imaju dodatna svojstva definirana relacijskom teorijom. Zbog toga se tablice u relacijskoj bazi podataka također nazivaju relacije.

Prema relacijskoj teoriji, svaka relacijska tablica sadrži dva obavezna dijela: **zaglavlje tablice** i **tijelo tablice**. Zaglavlje relacijske tablice sadrži skup atributa. Atributi predstavljaju svojstva koja su karakteristična za svaku entorku podataka u tablici. Tako npr., ako uzmemo tablicu „Studenti“ možemo joj pridružiti skup atributa: (JMBAG, OIB, Ime, Prezime, Datum\_rođenja, Mjesto\_rođenja). Svakom atributu odgovara određeni podatkovni tip, odnosno domena mogućih vrijednosti.

Tijelo relacijske tablice je zapravo skup entorki podataka (u tablici su to redovi osim zaglavlja). Entorka se sastoji od jedne ili više vrijednosti - podataka (skup vrijednosti). Svaka vrijednost pripada jednom atributu. Drugim riječima, svaka entorka se sastoji skupa po jedne vrijednosti (podatka) za svaki atribut. I zaglavlje, i tijelo relacije, kao i svaka entorka, su skupovi koji se sastoje od neuređenih elemenata. Svojstvo neuređenosti znači da niti u jednom od tih skupova nije moguće reći koji je njegov element prvi, drugi, treći itd. To svojstvo proizlazi iz relacijske teorije, ali u praksi mnogi DBMS-ovi ga se ne drže striktno, čak i SQL standard odstupa od njega.

U semantičkom smislu, svaka n-torka predstavlja neki iskaz u vezi područja interesa na koje se odnosi baza podataka. Svaka relacijska tablica predstavlja skup izjava odnosno iskaza, koji proizlaze iz entorke kao skupa vrijednosti (podataka) te skupa atributa relacije kojoj entorka pripada. Tako u spomenutoj relacijskoj tablici „Osobe“, entorka vrijednosti podataka:

*(1234567890, 0123456789012, Ante, Anić, 30.09.1989, Zagreb)*

se može interpretirati kao iskaz:

***Student s JMBAG-om: „1234567890“ i OIB-om „0123456789012“, se zove „Ante“, preziva „Anić“, rođen je „30. rujna 1989. godine“ u „Zagrebu“.***

Kao što vidimo, podaci u ovoj entorci mogu biti različitoga tipa. Neki predstavljaju brojkve, neki tekst, a neki datume. Ovdje smo naveli samo neke generičke tipove. U bazama podataka koristimo specifičnije tipove podataka koji su standardizirani na razini pojedinog DBMS-a, a u izvjesnoj mjeri i općenito prisutni u raspoloživim DBMS – sustavima (uz male iznimke). Može se postaviti pitanje zbog čega je to potrebno.

Prvo, osim što tip definira/ograničava moguće vrijednosti (domenu vrijednosti), on definira/ograničava i moguće operacije nad vrijednostima tipa.

Drugo, tipovi podataka omogućuju racionalnu upotrebu memorijskog prostora. Npr., broj 120 je kao tekst moguće zapisati u 3 bajta, dok ga je kao integer (cjelobrojni tip) moguće zapisati u 1 bajt memorije. Dakle, tip ne definira samo moguće operacije, sintaksna i semantička svojstva podatka, nego i način korištenja memorije prilikom njegova digitalnog zapisa.

Treće, pridruženi tip predstavlja inherentno ograničenje atributa na moguće vrijednosti odnosno znakove zapisa. Npr., ako je atributu pridružen tip integer, onda nije moguće njegovoj vrijednosti dodijeliti tekst. Ovo ograničenje ima semantičku, ali i sintaksnu dimenziju. Nadalje primijetimo, da svojstvo tipa proizlazi iz atributa, odnosno svaka vrijednost (podatak) u svakoj entorci nasljeđuje tip od atributa koji predstavlja. To povlači za sobom uvjet da su sve vrijednosti za isti atribut u svim entorkama homogene u pogledu tipa.

Važno je napomenuti da svaka relacija odnosno relacijska tablica mora imati jedinstveno ime (naziv) unutar baze podataka. Access dozvoljava da naziv tablice čini više riječi međusobno odvojenih prazninom, ali to nije dobra praksa, zbog toga što može dovesti do sintaksnih problema, a i zahtijeva posebnu pažnju u upotrebi. Ako se naziv tablice sastoji od više riječi najbolje ih je spojiti ili riječi međusobno razdvojiti s „\_“. Npr., tablica „*Položeni ispiti*“ se može nazvati „*Položenispiti*“ ili „*Položeni\_ispiti*“.

U Većini DBMS-ova pa tako i u Accessu, pozicije u entorkama (elementi) u koje se u tablicama upisuju podaci, jednostavno se zovu polja (Fields). Također se stupci tablice nazivaju kolone (Column). Isto tako se atribut naziva ime polja (Field Name). Entorka relacije se također naziva red (Row).

## 2.4.2 Tipovi podataka

U Ms Accessu postoji nekoliko tipova unaprijed definiranih podataka. Neki DBMS sustavi omogućuju i tzv. korisnički definirani tip podatka, dok Access to na razini DBMS-a ne omogućuje. Postoje osnovni tipovi polja i njihovi podtipovi. Neki tipovi imaju podtipove, dok drugi nemaju.

Osnovni tipovi u Accessu su:

- Number** - numerički koji može sadržavati nekoliko podtipova numeričkih podataka uključujući cjelobrojne tipove kao i decimalne tipove
- Text** - tekstualni varijabilne duljine do 255 znakova
- Memo** - tekst sa preko 255 znakova
- Date/time** - datum/vrijeme
- Yes/No** - logički tip koji može sadržavati jednu od dvije logičke vrijednosti: istina/laž odnosno true/false ili yes/no
- Currency** - prikaz novčanih vrijednosti (decimalna vrijednost sa oznakom valute)
- Hyperlink** - hiperlink veza ili HTTP adresa

**Autonumber** - predstavlja automatski generator rastućih pozitivnih cijelih brojeva – svaki puta kada se ubaci nova entorka podataka u tablicu, u autonumber polju će se generirati nova veća vrijednost (za 1 veća od najveće upisane za to polje) – nije ga moguće mijenjati niti brisati

**OLE Objekt** - oznaka OLE objekta (slika, datoteka,...)

Numerički tip mora imati definiran podtip i broj decimalnih mjesta zapisa. Postoje tri cjelobrojna podtipa: **Byte**, **Integer**, **Long integer**. Oni se međusobno razlikuju prema broju bajtova koji troše u memorijskom zapisu: 1, 2 ili 4 Byte. Iz toga proizlazi maksimalna numerička vrijednost koji mogu sadržavati:  $2^8$ ,  $2^{16}$ ,  $2^{32}$ .

Zatim, tu je podtip koji omogućuje zapis s kliznim zarezom u jednostrukoj (**Single**) i dvostrukoj (**Double**) preciznosti. **Decimal** omogućuje po volji oblik zapisa decimalnih brojeva uz fiksnu poziciju decimalnog zareza. Currency tip je poseban tip, iako je u semantičkom smislu numerički tip, karakteriziran s dva decimalna mjesta i posebnom oznakom valute prilikom ispisa na formi ili izvještaju.

### 2.4.3 Ključevi relacije

Rekli smo da relacijska tablica za razliku od obične tablice podataka, posjeduje posebna svojstva koja proizlaze iz definicije relacije u relacijskoj teoriji, odnosno u relacijskom modelu podataka. Pored već navedenih svojstava, tu je važno svojstvo jedinstvenosti entorki u pojedinoj relaciji. To znači da se entorke u relaciji ne smiju duplicirati. Pojam jedinstvenosti je složeniji nego što bi se to moglo shvatiti doslovno usporedbom entorki. Naime, jedinstvenost ne proizlazi iz cijelog skupa vrijednosti entorke, nego iz određenog podskupa vrijednosti (atributa), koji ne mora nužno biti njegov pravi podskup. Jedinstvenost entorke se može utvrditi na temelju tzv. ključa relacije odnosno kandidatnog ključa.

U relacijskoj teoriji podataka **pravilo kandidatnog ključa** je definirano na sljedeći način:

Za svaku relaciju **R**  $\{a_1, a_2, a_3, \dots, a_n\}$  mora postojati barem jedan neprazni podskup **P**  $\{a_1, a_2, a_3, \dots, a_x\}$ , pri čemu vrijedi:  $x \leq n$ , i  $n, x > 0$ . Svaki takav skup atributa se naziva **kandidatni ključ**, odnosno kandidatni ključevi kad ih ima više, pri čemu moraju biti ispunjena oba uvjeta:

- Skup vrijednosti podataka tog podskupa atributa jednoznačno identificira svaku entorku u danoj relaciji **R**,
- Niti jedan pravi podskup  $P_p \{a_1, a_2, a_3, \dots, a_y\}$  skupa **P**, nema to svojstvo za bilo koju entorku relacije **R**, odnosno ne postoji niti jedan  $P_p$  s takvim svojstvom ako je:  $y < x$ .

Svojstvo **a.** se naziva **svojstvo jedinstvenosti**, a svojstvo **b.** se naziva **svojstvo nesmanjivosti** ili **ireducibilnosti**. Budući da prema tom pravilu, ključeva relacije može biti više, u relacijskoj teoriji podataka govori se kandidatnim ključevima relacije. Dakle, ako ih je više, svi su oni kandidati za ključ, odnosno identifikatori relacije. Primijetimo da je u slučajevima kad se ključ relacije sastoji od samo jednog atributa, svojstvo ireducibilnosti uvijek ispunjeno, budući da je tada broj atributa minimalan.

Razmotrimo kao primjer navedenu relaciju „Student“. Relacija ima dva podskupa svojih atributa koji su kandidatni ključevi:  $\{JMBAG\}$  i  $\{OIB\}$ . JMBAG je za svakog studenta jedinstvena šifra unutar ISVU-sustava. OIB je jedinstvena šifra svakog građanina u RH, prema tome oba zadovoljavaju svojstvo jedinstvenosti, odnosno mogu jednoznačno identificirati svakog studenta u relaciji student. Budući da se oba sastoje od jednog jedinog atributa, oba zadovoljavaju i drugi uvjet - nesmanjivost. Međutim, ako bismo uzeli podskup  $\{JMBAG, OIB\}$  ili ako bi smo navedenim kandidatnim ključevima dodali još neki od atributa (bilo koji i bilo koliko njih, npr.:  $\{JMBAG, Ime, Prezime\}$  ili  $\{Datum\_rođenja, OIB\}$ ), svojstvo jedinstvenosti bi ostalo ispunjeno, ali svojstvo minimalnosti ne bi bilo zadovoljeno pa prema tome, to onda ne bi bili kandidatni ključevi.

U praktičnim slučajevima konkretnih DBMS-a, uvodi se pojam **primarni ključ**. Primarnim ključem se imenuje jedan od kandidatnih ključeva. Svaka relacijska tablica mora posjedovati primarni ključ (PK). Primarni ključ (Primary key) je istovremeno i kandidatni, razlika je samo u tome što u nekoj relaciji može postojati jedan ili više kandidatnih ključeva, a uvijek samo jedan primarni ključ. Primarni ključ iako je neizostavni dio strukture tablice, predstavlja i temeljno ograničenje relacije.

Prvo svojstvo znači da za svaku relacijsku tablicu trebamo odrediti neprazan skup atributa čije će vrijednosti biti jedinstvene, tj., iste vrijednosti primarnog ključa će se pojaviti u samo u jednoj entorci relacije. Pri tome u sastav primarnog ključa mogu ući i svi atributi relacije, u slučaju ako ne postoji neki manji podskup koji ima svojstvo jedinstvenosti.

Koncept kandidatnog ključa postoji samo u teoriji, a u praksi baza podataka odnosno DBMS-a, on je pojednostavljen kroz koncept primarnog ključa. Važno je naglasiti da koncept kandidatnog ali primarnog ključa ima prije svega semantičko značenje. Jednoznačnost kao svojstvo ključa relacije proizlazi iz vrijednosti prirodnih osobina objekata koje relacija opisuje. Svaka entorka je iskaz/izjava u vezi nekog objekta iz područja interesa za koji baza podataka predstavlja model. Ti se objekti moraju moći jednoznačno identificirati preko svojih osobina/atributa. Relacijske tablice nisu objekti u bazi u koje ima smisla upisivati bilo kakve podatke.

Neki projektanti baze podataka jednostavno definiraju tzv. **surogatne ključeve**. Ključ-surogat je atribut koji nema drugo značenje, osim da jednoznačno identificira entorku odnosno red u tablici, on ni na koji način ne identificira modelirani objekt, jer njegova semantika ne postoji izvan baze podataka. Surogati se koriste najčešće zbog poboljšanja performansi kod izvođenja SQL upita i transakcija. Čak i kad se koriste za tu svrhu, što može biti u nekim slučajevima prihvatljivo, oni ne mogu zamijeniti prirodne ključeve. Ograničenje koje proizlazi iz prirodnih kandidatnih/primarnih ključeva ne može biti zamijenjeno surogatima ključa, jer surogati ne sadrže nikakvo semantičko značenje niti ograničenje.

U Accessu ćemo primarni ključ pridružiti tablici tako da selektiramo sve one attribute za koje želimo da čine primarni ključ i dodijelimo im primarni ključ pritiskom na desnu tipku miša (Primary key).



Ako to ne učinimo Access će nas kod spremanja tablice upozoriti da ne postoji PK, te će nam ponuditi automatski autonumber kao ključ (što nije dobro – PK treba uvijek definirati). Primarni ključ se može dodijeliti odnosno promijeniti i naknadno, ali ako su relaciju upisane neke entorke, koje nemaju svojstvo jedinstvenosti (nedupliciranja), nećemo moći promijeniti PK dok ne promijenimo ili izbrišemo takve entorke.

#### 2.4.4 Ograničenja i formati na razini atributa

Osim tipa i podtipa, svakom atributu možemo dodijeliti i neka druga svojstva. Najvažnija od tih svojstava su ograničenja :

**Default vrijednost** - to je pretpostavljena uobičajena vrijednost koja će biti ponuđena prilikom ubacivanja nove entorke, uvijek možemo promijeniti ponuđenu (default-vrijednost).

**Validation rule** - to je pravilo koje će se provjeravati/primjeniti na novo unesenu ili naknadno promijenjenu vrijednost polja. Ako nije ispunjen uvjet koje pravilo definira, neće se prihvatiti unesena vrijednost polja.

**Validation text** - to je poruka o neispravnosti validacije koja će se prikazati ako prethodno pravilo nije ispunjeno.

**Required** - ako je postavljeno ovo svojstvo na Yes, prilikom ubacivanja entorke u to polje mora biti upisana vrijednost (ne može se preskočiti) - to pravilo u tom slučaju zabranjuje da polje poprimi tzv. null-vrijednost.

**Allow zero Length** - Ako je postavljeno na NO u polje mora biti upisan barem jedan znak. Prilikom upisa polja odnosno ubacivanja entorke, provjeravat će se podudarnost s ovim zahtjevima pa ako ne budu ispunjeni, neće se prihvatiti započeta akcija izmjene odnosno ubacivanja entorke.

**Indexed** - odnosi se na kreiranje indeksa nad poljem radi bržeg pristupa ili radi osiguranja jedinstvenosti polja koje nije primarni ključ. Uočite da će se prilikom definiranja primarnog ključa nad jednim poljem polju dodijeliti Indeks=(No duplicates). U drugim bazama se jedinstvenost neključnog polja postiže klauzulom Unique.

**Format** - definira oblik u kojem će defaultno biti prikazivani podaci polja.

**Caption** - predstavlja alternativni naziv polja koji će biti ispisan na Access formama.

**Input mask** - omogućuje defaultni/pretpostavljeni format kod unosa polja na formama.

Svojstva polja (atributa) Format, Caption, Input mask, Unicode Compression i ostala, odnose se na način prikaza i zapisa polja. Osim Unicode compression koji znači optimizaciju Unicode zapisa tekst polja (racionalizaciju na jedan bajt kad je to moguće), ostala svojstva su karakteristična za Access.

## 2.5 Sadržaj i tijek izvođenja vježbe

### 2.5.1 Sadržaj vježbe

U ovoj vježbi je potrebno kreirati novu bazu **Vj-2-2013.accdb** koja će sadržavati nekoliko relacijskih tablica, a predstavljat će pojednostavljeni dio baze iz prethodne vježbe.

Tablica **Artikl** sadrži atribute (naziv, opis):

Field Name	
IdArtikla	šifra artikla
Naziv	naziv artikla
Jmj	standardna jedinica mjere
Cijena	prodajna cijena
GrupaArt	klasifikacijska šifra artikla
Dobavljač	id standardnog dobavljača/proizvodjača artikla

Tablica sadrži podatke o artiklima koje zamišljeno poduzeće prodaje/kupuje

Tablica **PPartner** sadrži atribute (naziv, opis):

Field Name	
IdPP	id poslovnog partnera (kupca/dobavljača)
Naziv	naziv poslovnog partnera
Mjesto	sjedište u mjestu PP
Adresa	adresa sjedišta PP

Tablica sadrži podatke o poslovnim partnerima s kojima se posluje (dobavljači/kupci)

Tablica **Promet** sadrži atribute (naziv, opis):

Field Name	
PGod	poslovna godina
TipDok	tip dokumenta
BrojDok	broj dokumenta
IdArtikla	šifra artikla
Jmj	jedinica mjere
Cijena	cijena za 1mj
UKoličina	količina ulaza u skladište
IKoličina	Količina izlaza iz skladišta
DatumUpisa	datum kada je podatak upisan

Tablica sadrži podatke o prometu artikala (ulaz/izlaz) prema određenim tipovima dokumenata

Tablica **TipDok** sadrži atribute (naziv, opis):

Field Name		D
TipDokProm	skraćena oznaka za tip/vrsta dokumenta prometa	
Naziv	naziv/opis tipa dokumenta	
UI	oznaka za: U/I/N-dokument je ulazni/izlazni/nije prometni dokument	

Tablica sadrži podatke o tipovima dokumenata s oznakom vrste prometa

Za svaku od ovih tablica kreirajte tablicu u bazi s odgovarajućim tipom polja, ograničenjima i primarnim ključem. U Accessu se tablice kreiraju putem izbornika: **Create/Table Design**. Vodite računa da polja s istim značenjem koja se nalaze u više tablica (bez obzira na naziv) imaju isti tip i podtip. Koja su to polja?

## 2.5.2 Tijek izvođenja vježbe

1. Kreirajte novu bazu Vj-2-2013.accdb
2. Kreirajte tablicu Artikl
3. Kreirajte tablicu PPartner
4. Kreirajte tablicu TipDok
5. Kreirajte tablicu Promet
6. Provjerite logičnost tipova polja i postavljenih ograničenja
7. Provjerite logičnost i ispravnost definiranih primarnih ključeva – prodiskutirajte rješenja s kolegama: ako uočite greške ispravite ih.
8. Upišite podatke prema vlastitom nahođenju (prisjetite se prethodne vježbe)
9. Provjerite da li možete unositi podatke na logičan način, jesu li ključevi ispravno definirani ili predstavljaju neprirodno ograničenje kod unosa/promjene podataka
10. Ako ste uočili bili kakve probleme u točki 9, posavjetujte se s kolegama, tražite pomoć asistenta.

## 2.6 Završna pitanja i zadaci

### 2.6.1 Završna pitanja

Po završetku vježbe, studenti bi trebali bi znati ispravno odgovoriti na slijedeća pitanja (test će se bazirati na ovim pitanjima):

- Što je relacija, relacijska tablica?
- Koji su njeni dijelovi?
- Može li relacija sadržavati dva ista atributa?
- Može li relacija imati nula entorki (biti prazna)?
- Kakva je razlika između tipa i podtipa podatka?
- Koji su glavni tipovi i podtipovi mogući u Accessu?

- Zbog čega je tip podatka bitan?
- Što tip određuje/ograničava?
- Koja su ograničenja nad atributima moguća u Accessu?
- Kakvo je njihovo značenje?
- Što je to validacijsko pravilo atributa (Validation rule)
- Koja su svojstva primarnoga ključa?
- Koja je razlika između kandidatnog i primarnog ključa?
- Koliko kandidatnih ključeva može imati relacija?
- Može li/mora li tablica koja nema niti jednu entorku imati primarni ključ?
- Koliko primarnih ključeva ima svaka tablica/relacija?
- Što predstavlja entorka?
- Od čega se entorka sastoji?
- Koji je najmanji element/objekt u bazi podataka koji sadrži elementarni podatak?
- Kako se baza ponaša ako pokušamo unijeti podatak koji ne odgovara postavljenim ograničenjima?
- Vrijedi li isto u slučaju kada pokušamo podatak promijeniti?

### 2.6.2 Zadaci za samostalno produblivanje znanja

- Nakon vježbe diskutirajte s kolegama oko rješenja vježbe. Diskutirajte na temelju argumenata temeljenih na uvodu u vježbu te na temelju predavanja iz kolegija baze podataka.
- Razmislite koje bi još atribute imalo smisla dodati u navedeni primjer, pokušajte to učiniti
- Koje bi se tablice mogle dodati? Konzultirajte se s nastavnikom ako ste u dilemi.

## 2.7 Literatura i dodatni materijali

Detalje o tipovima podataka u Ms Accessu 2010, obavezno proučite na Microsoftovoj stranici:

<http://office.microsoft.com/hr-hr/access-help/uvod-u-vrste-podataka-i-svojstva-polja-HA010341783.aspx>

## Vježba 3: Query, pogledi, ograničenja

### 3.1 Motivacija

U prethodnoj smo vježbi obradili koncept ograničenja na razini atributa relacijske tablice. U ovoj ćemo vježbi nastaviti s implementacijom ograničenja. Tema ove vježbe je referencijalno ograničenje ili tzv. referencijalni integritet, koji proizlazi iz koncepta stranog ključa. Strani ključ i referencijalni integritet su osobito bitni koncepti kako u teorijskom, tako i u praktičnom smislu, kod transakcijskih relacijskih baza podataka. Pored toga, obradit ćemo i neka druga ograničenja na razini atributa te na kraju i validacijske poglede. Pri tom ćemo koristiti Accessov Query alat kao jednostavan, ali efikasan Query by Example tabelarni jezik za izradu SQL upita, bez potrebe da se pritom poznaje SQL jezik.

### Sadržaj vježbe:

<b>VJEŽBA 3:</b>	<b>QUERY, POGLEDI, OGRANIČENJA</b>	<b>41</b>
3.1	MOTIVACIJA	41
3.2	CILJEVI VJEŽBE – ISHODI UČENJA	42
3.3	SAMOSTALNA PRIPREMA VJEŽBE	42
3.4	TEORIJSKA PRIPREMA VJEŽBE	43
3.4.1	<i>Dodatna ograničenja atributa</i>	43
3.4.2	<i>Strani ključ</i>	45
3.4.3	<i>Uspostavljanje referencijalne veze</i>	46
3.4.4	<i>QBE - Query</i>	48
3.4.5	<i>Validacijski pogledi</i>	49
3.5	ZADACI I SADRŽAJ VJEŽBE	51
3.5.1	<i>Sadržaj vježbe</i>	51
3.5.2	<i>Tijek izvođenja vježbe</i>	52
3.6	ZAVRŠNA PITANJA U VEZI IZLOŽENE VJEŽBE	53
3.7	ZADACI ZA SAMOSTALNO PRODUBLJIVANJE ZNANJA	54
3.8	LITERATURA I DODATNI MATERIJALI	54

## 3.2 Ciljevi vježbe – ishodi učenja

Od studenata se očekuje da, nakon uspješno završene vježbe:

- Shvate pojam eksplicitnih ograničenja teorijski i praktično
- Shvate pojam i ulogu validacijskih pogleda
- Nauče izraditi jednostavan pogled pomoću alata Query
- Shvate značenje i primjenu stranih ključeva

## 3.3 Samostalna priprema vježbe

Prije dolaska u lab, obaveze studenata su:

- Proučite prethodnu materiju sa predavanja iz kolegija BP
- Utvrdite prethodnu vježbu – pitanja u prethodnoj vježbi će predstavljati temelj za blic-test u ovoj vježbi
- Proučite pripremne materijale za ovu vježbu
- Proučite sadržaj vježbe –pokušajte razumjeti izloženo
- Minimalno potrebna znanja za pristupanje vježbi:
  - Koja su dodatna ograničenja moguća osim ograničenja navedenih u prethodnoj vježbi?
  - Što znači NotNull i kako se specificira?
  - Što je to strani ključ?
  - Kako glasi definicija referencijalnog integriteta – kakvo ima značenje?
  - Što su kaskadne operacije i zbog čega su potrebne?
  - Što je QBE i kako je realiziran u Accessu?

## 3.4 Teorijska priprema vježbe

### 3.4.1 Dodatna ograničenja atributa

Temeljna ograničenja atributa u relacijskoj bazi podataka, a to vrijedi i za Access, su ograničenja tipa. To znači da će svakom atributu kod kreiranja biti dodijeljen tip podataka, odnosno njegov tip. Prisjetimo se da se takva pravila zovu ograničenja zbog toga što će njihovo provođenje osigurati DBMS, prilikom operacija nad podacima u bazi (ubacivanje, mijenjanje, ...) – dakle moguće vrijednosti atributa za svaku entorku relacije bit će ograničene definicijom tipa. Prisjetimo se također iz prethodne vježbe da pored ovih, Access omogućuje još definiranje sljedećih ograničenja atributa: Validacijsko, Not-Null i Unique.

**Validacijsko** ograničenje (Validation svojstvo) se definira kao aritmetičko logički izraz koji uvijek daje **logičku vrijednost (logički tip)**. On se izračunava na temelju vrijednosti atributa (polja) i prema tome, može dati jednu od dvije vrijednosti:

- **Istina** (True), ili
- **Laž** (False)

Ako izraz daje logičku vrijednost „istina“ (True), vrijednost će se prihvatiti/dopustiti. U suprotnom ako izraz daje logičku vrijednost „laž“ (False), DBMS neće dopustiti upis vrijednosti polja za dani atribut odnosno entorku. Validacijska ograničenja pripadaju u eksplicitna ograničenja atributa.

***Primjer:** Ako u tablici *Artikl* iz prethodne (i ove) vježbe želimo ograničiti vrijednosti atributa *cijena* na sve pozitivne vrijednosti veće od 0 i manje 100000, definirat ćemo Validtion:  $> 0 \text{ AND } >100000$ . Za validation text (to je poruka koju će DBMS uputiti ako se ograničenje pokuša prekršiti) mogli bi staviti: „Cijena mora biti veća od nule, a manja od 100000!“.*

*Uloga teksta poruke je ovdje važna, jer će korisniku pojasniti zbog čega unesena/promijenjena vrijednost nije prihvaćena. Nedopustiva je praksa ostaviti korisnika u neznanju zbog čega podatak nije prihvaćen.*

**NOT-NULL** ograničenje se u Accessu uspostavlja s vrijednošću svojstva Required. To svojstvo može imati jednu od dvije vrijednosti: True/False (istina/laž).

Ako je postavljeno na „True“, ograničenje garantira da će se obavezno morati ubaciti odgovarajuća vrijednost u polje u trenutku ubacivanja cijele entorke u bazu podataka. Dakle, u tablici neće moći biti niti jedna entorka koja u polju za dani atribut nema unesenu valjanu vrijednost (u okviru zadanoga tipa i ostalih ograničenja).

Ako je ograničenje Required postavljeno na vrijednost „False“, neće biti nužno upisati vrijednost u svaku entorku za atribut za koji je ograničenje definirano. Dakle, moći će postojati entorke koje za dani atribut nemaju nikakvu vrijednost. Uobičajeno je da se kaže kako je ta vrijednost: „Null“ (ništavna,

nepostojeća). Međutim, to je zapravo samo oznaka da vrijednost ne postoji, to nije vrijednost poput ostalih vrijednosti.

Nul-vrijednost ima posebno značenje u bazama podataka. Ona nije trivijalna. Prvo, operacije nad podacima koji sadrže null-vrijednost ne podliježu istim pravilima koja vrijede za ostale vrijednosti.

Drugo, u semantičkom smislu nul-vrijednost može se odnositi na jedan od dva slučaja:

**Vrijednost je nepoznata** ili nedostupna – u trenutku upisa entorke vrijednost atributa za tu entorku nije poznata, ali u semantičkom smislu nije egzistencijalno bitna. Vrijednost se može upisati naknadno kad bude poznata (ako bude potrebno). U ovom slučaju radi se o podacima koji su mogući, ali nisu nužni.

*Npr., u tablici Dokument u ovoj vježbi atribut Opis predstavlja neformalni podatak-napomenu uz dokument u obliku slobodnog opcionalnog teksta. Ako kod nekog dokumenta nema potrebe za takvim opisom, polje Opis u toj entrci može poprimiti nul-vrijednost.*

**Vrijednost je neprimjenjiva** na danu entorku – pojavljuje se u slučaju kada bi vrijednost atributa za neku entorku bila besmislena, odnosno nedozvoljena i bez smisla. Ako u ovom slučaju polje jednom poprimi nul vrijednost, ono će uvijek imati nul-vrijednost (osim ako se ne promijeni semantičko značenje tablice i odgovarajućeg atributa). Semantički, ovaj oblik nul-vrijednosti najvjerojatnije ukazuje da tablica nije dobro osmišljena – atributi koji su neprimjenjivi na neke od entorki ukazuju da entorke koje mogu imati vrijednost u takvoj relaciji, treba specijalizirati u novu relaciju koja će osim specijaliziranog atributa imati i primarni ključ isti kao i polazna relacija, pri čemu će taj primarni ključ u specijaliziranoj relaciji biti istovremeno strani ključ u toj relaciji. O tome će biti više riječi u materijalima o oblikovanju baze podataka (u sljedećim vježbama).

*Npr., kod relacije Ppartner mogli bi smo dodati atribut „Predsjednik uprave“. Međutim, to bi bilo primjenjivo samo za dio poslovnih partnera. Tako npr., za poslovne partnere pojedince/obrte taj bi atribut bio neprimjenjiv, pa bi u takvim entorkama trajno poprimio nul-vrijednost. Kao što smo napomenuli, to bi ukazivalo na upitnu konstrukciju tablice pa bi u konačnici zahtijevalo dodatnu novu tablicu, ali detaljnija rasprava o tome prelazi cilj i opseg ove vježbe.*

**Unique** (jedinствена vrijednost) ograničenje se odnosi na zahtjev da za svaku entorku vrijednost tog atributa bude jedinstvena odnosno da je jednoznačna za cijelu tablicu. Dakle, ne smije se pojaviti dvije ili više entorki u relaciji za koje bi vrijednost tog atributa bila ista – sve moraju biti različite. U vezi s tim, važno je naglasiti dvije stvari. Prvo, svojstvo/ograničenje Unique se odnosi na nepravi skup atributa, dakle, skup može sadržavati jedan ili više atributa koji kao cjelina moraju imati ovo svojstvo. Drugo, ovo svojstvo/ograničenje je inherentno (neizostavno svojstveno) za svaki primarni ključ (svaki primarni ključ mora imati svojstvo jedinstvenosti). Međutim, prema drugom temeljnom svojstvu primarnog ključa (nesmanjivost ili ireducibilnost), jedinstvenost ne smije biti ograničenje na bilo koji dio primarnog ključa.



To istovremeno znači da niti jedan dio primarnog ključa (niti jedan njegov atribut) ne smije poprimiti nul-vrijednost.

U alatima Accessa se jedinstvenost definira kao indeks te može poprimiti vrijednosti:

- **Indexed: Yes (No Duplicates)** – indeks je postavljen nad atributom bez dupliciranja. To znači, istovremeno je postavljen indeks i ograničenje jednoznačnosti (Unique).
- **Indexed: Yes (Duplicates Ok)** – indeks je postavljen, ali jednoznačnost nije osigurana – dozvoljeni su duplikati vrijednosti dotičnoga atributa u različitim entorkama.
- **Indexed: No** – nije postavljen indeks i nije osigurana jednoznačnost.

Ovdje moramo napomenuti da indeksiranje pripada u domenu fizičkog oblikovanja baze podataka. Naime indeksi osiguravaju brzi (random) pristup entorkama na osnovu zadane vrijednosti, te tako imaju neposredni utjecaj na performanse izvođenja operacija u bazi podataka. Budući da bi performanse baze bile dramatično smanjene kad primarni ključ ne bi imao direktni pristup, on se automatski u Accessu postavlja nad primarnim ključem. Postoji više drugih metoda ubrzanja pristupa i indeksiranja, ali to također prelazi opseg i cilj ove vježbe i pripada u područje fizičkog oblikovanja baze podataka.

Napomenimo da način kreiranja i pregledavanja tablica /relacija, atributa i ograničenja u Accessu koji smo do sada koristili, nije karakteristika DBMS-a općenito, nego je to specifičnost Accessa izvedena kao posebna tabelarna forma/alat. Ta funkcionalnost Accessa ide u prilog lakšeg praktičnog savladavanja osnovnih koncepata baza podataka – upravo zbog toga ju ovdje koristimo. Standardni način kreiranja baze podataka je pomoću SQL jezika. On omogućuje precizniji, ali i složeniji način specifikacije sheme baze podataka. O tome više u idućim vježbama.

### 3.4.2 Strani ključ

Svaka relacijska tablica mora posjedovati primarni ključ (**PK**). Ako se neki atribut koji je u jednoj relaciji primarni ključ, pojavljuje u toj istoj relaciji ili nekoj drugoj relaciji kao neki drugi atribut, onda kažemo da je on u ulozi **stranog ključa (Foreign Key)**. Neki puta se strani ključ naziva vanjski ključ, ali strani ključ je po našem mišljenju primjerenije. Budući da se primarni ključ može sastojati i od više atributa, onda to isto vrijedi za strani ključ. Za svaku tablicu postoji samo jedan primarni ključ, dok u istoj tablici može postojati više stranih ključeva.

Značaj stranog ključa je višestruk:

- On osigurava logičko povezivanje podataka koji imaju isto semantičko značenje. To je povezivanje moguće između dvije relacije, ali i unutar iste relacije.
- On doprinosi smanjenju redundancije podataka (zalihosti, višestrukog zapisa istih podataka).
- On ima neposredan utjecaj na održanje konzistencije podataka u bazi podataka.

Strani ključ predstavlja također jedan specifičan oblik ograničenja. Radi se o jednom od temeljnih ograničenja u bazi podataka o tzv., pravilu referencijalnog integriteta .

**Pravilo referencijalnog integriteta** (referential integrity rule): „U svakom trenutku, za svaku vrijednost stranoga ključa mora postojati takva ista vrijednost u odgovarajućoj (referentnoj) tablici u kojoj je to vrijednost primarnog ključa.“

Kaže se da strani ključ povezuje tablice, ali to povezivanje je zapravo utemeljeno na povezivanju entorke. Pri tom se referentna tablica označava kao tablica-roditelj (parent-table), a referencirana tablica se označava (naziva) tablica-dijete (child-table).

*Primjer: Uzmimo relacije Dokument i Radnik. Relacija radnik ima primarni ključ IdZap (šifra ili identifikator zaposlenika/radnika). On je u tablici radnik primarni ključ pa stoga njegove vrijednosti jednoznačno identificiraju svaku entorku u tablici Radnik. Tablica Dokument predstavlja/sadrži osnovne podatke o izdanim dokumentima. Svaki dokument u toj tablici je kreirao odnosno izdao neki radnik. Atribut Izdao predstavlja šifru/identifikator radnika (vrijednost IdZap) koji je izdao dokument. Semantički gledano, atribut Izdao može poprimiti samo vrijednosti atributa IdZap iz tablice Radnik. Dakle, za svaku vrijednost Izdao mora postojati odgovarajuća vrijednost IdZap u tablici Radnik. Prema tome, možemo zaključiti da su tablice semantički/logički povezane. Ta veza je oblik referencijalne veza, u kojoj je atribut Izdao strani ključ pa se stoga na njega mora primijeniti pravilo referencijalnog integriteta.*

### 3.4.3 Uspostavljanje referencijalne veze

Uspostavljanje referencijalne veze odnosno stranog ključa, predstavlja zabranu/restrikciju narušavanja referencijalnog integriteta. To znači da će DBMS spriječiti one aktivnosti nad podacima u bazi koje bi izazvale narušavanje referencijalnog integriteta. Međutim, dinamička priroda podataka u bazi zahtijeva da se neki puta izvrše akcije nad podacima koje bi bazu mogle dovesti u stanje narušenoga integriteta. Budući da je takvo stanje baze nedozvoljeno, DBMS posjeduje mehanizme koji osiguravaju automatsku korekciju podataka s ciljem uspostavljanja integriteta. Ti automatski mehanizmi se uspostavljaju kod akcija na koje su postavljeni (ako su postavljeni).

Tipični su sljedeći mehanizmi uspostavljanja referencijalnog integriteta podataka u bazi:

**Restrikcija** – DBMS će odbiti izvršavanje promjene nad podacima ako bi to izvršavanje narušilo referencijalni integritet – nikakva druga akcija se neće pokrenuti.

**Kaskadno ažuriranje** – prilikom ažuriranja ( promjene vrijednosti) vrijednosti primarnog ključa u nekoj entorci u tablici-roditelj, izvršit će se automatsko ažuriranje (ispravljanje) vrijednosti stranog ključa u tablici-dijete za sve pripadajuće (referencirane) entorke te će se na taj način osigurati referencijalni integritet.

**Kaskadno brisanje** – prilikom brisanja entorke u tablici-roditelj, izvršit će se automatsko brisanje svih referenciranih entorki u tablici-dijete, te će se na taj način osigurati referencijalni integritet.

**Nulifikacija** – promjena/brisanje vrijednosti primarnog ključa u tablici –roditelj izazvat će postavljanje stranog ključa na nul-vrijednost u odgovarajućim entorkama referencirane tablice.

Kaskadne operacije se zovu kaskadne zbog toga što će se njihovo izvršavanje automatski prenijeti na nižu razinu tablica. To znači ako se kaskadna operacija izvrši zbog promjene vrijednosti primarnog ključa u nekoj tablici –roditelj, nad nekom tablicom-dijete koja je u odnosu neku treću tablicu u ulozi tablice-roditelj, automatska akcija će se primijeniti na treću tablicu i rekurzivno dalje na sve podređene razine sve dok se ne uspostavi referencijalni integritet.

Semantički nije svejedno koji ćemo mehanizam definirati/postaviti kod koje veze. Moramo znati da to neposredno ima bitan utjecaj na semantičku reprezentaciju modeliranog područja interesa. Detaljnije objašnjenje prelazi okvir ove vježbe, jer se odnosi na konceptualno odnosno logičko modeliranje podataka. U ovoj vježbi nam je cilj ukazati da ti mehanizmi postoje te pokazati njihovu svrhu i posljedice djelovanja.

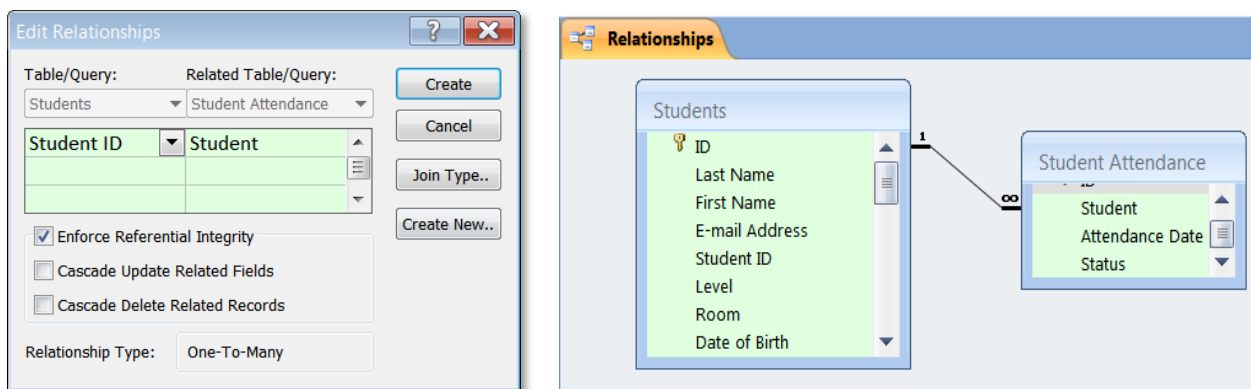
U Accessu je moguće uspostaviti prva tri mehanizma, dok nulifikaciju nije moguće definirati kao automatsku akciju. Za tu svrhu u Accessu postoji alat koji se zove „Relationship“. Radi se o formi koja se aktivira kroz meni: **Database Tools -> Relationship**

Nakon izbora, prikazat će se forma na kojoj će grafički prikazane tablice sa pripadajućim atributima uključujući i oznaku primarnih ključeva (ikona „ključić“ ispred atributa) i referencijalne veze među njima. U početku će to biti prazna forma. Postupak definiranja stranog ključa je sljedeći:

Pritiskom na desnu tipku miša u okviru forme „Relationship“ i to na prazni prostor pojavit će se skočni izbornik (Pop-up) u kojemu odaberemo „Show Table...“

Nakon što se prikaže prozor „Show Table“ odaberemo Tables-tab te izaberemo jednu ili više tablica i pritisnemo tipku „Add“. Ovaj postupak možemo ponavljati dok ne dobijemo grafički prikaz svih potrebnih tablica.

U sljedećem koraku povezujemo attribute dviju tablica između kojih želimo uspostaviti referencijalni integritet. Najprije odaberemo atribut koji je primarni ključ te ga sa pritisnutim mišem povežemo atributom koji treba biti strani ključ. Ako se strani ključ (isto i primarni) sastoji od više atributa, učinimo isto povezivanje za svaki od atributa koji ga čini ili dodajmo sve atributa PK/SK u formu „Edit Relationships“.



Restrikciju (uspostavljanje referencijalnog integriteta), te uključivanje kaskadnih operacija postizemo označavanjem ponuđenih opcija na formi.

Ako želimo vezu izbrisati, označimo liniju veze i pritisnemo „Delete“.

Ako želimo vezu izmijeniti pozicioniramo se na liniju veze i pritisnemo dvostruki klik.

Access omogućuje uspostavljanje referencijalnog integriteta i nad virtualnim/izvedenim relacijama (u Accessu se zove Query).

Važno je napomenuti da referencijsku vezu (odnosno ograničenje stranoga ključa) nećemo moći uspostaviti ako su podaci uneseni u tablice koje povezujemo takvi da ne zadovoljavaju referencijalni integritet. To će se uvijek dogoditi ako podaci u ključu tablice-dijete ne postoje u u primarnom ključu tablice-roditelj. Također je potrebno da su atributi koje povezujemo u cijelosti u sastavu primarnoga ključa. Ako se dogodi koji od ovih problema prilikom uspostavljanja ograničenja stranog ključa, onda najprije treba izbrisati podatke koji su u vezi pa tek nakon toga definirati ograničenje stranoga ključa.

### 3.4.4 QBE - Query

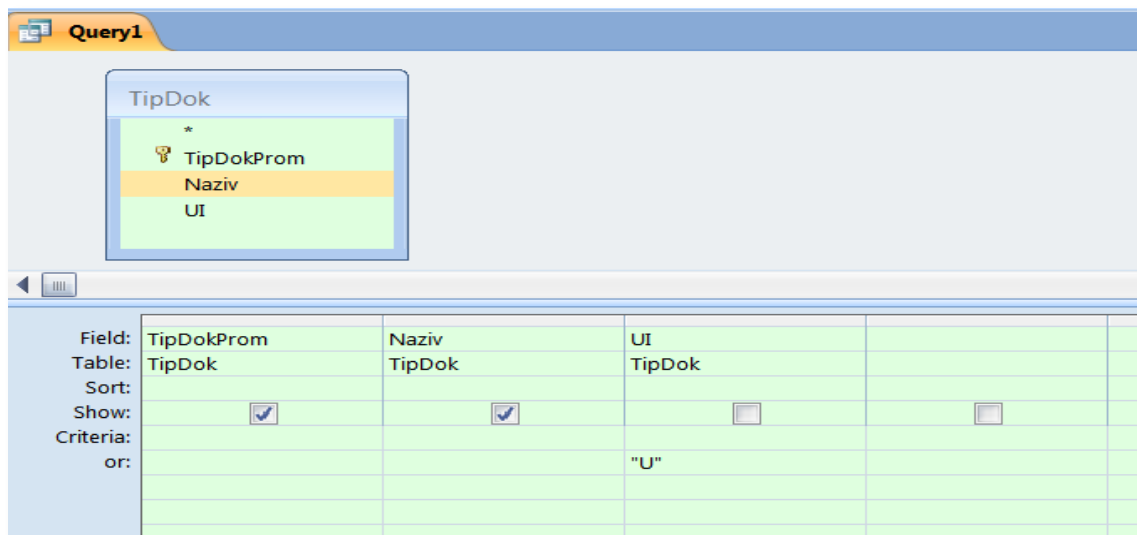
Dosadašnji koncept relacijskih tablica koji smo objašnjavali odnosi se na tzv. **bazne relacije**. Pored njih u relacijskoj bazi postoje i tzv., **izvedene ili virtualne relacije**. Često se nazivaju **pogledi (View)**. Pogledi su u bazi definirane relacije koje u trenutku kad im se pristupi (svaki puta) izvrše generiranje virtualne relacije. Podaci koje pogled generira se nalaze u baznim tablicama. Pogled može biti definiran nad jednom ili više baznih relacija, nad jednom ili više drugih pogleda te u bilo kojoj kombinaciji jednog i drugog. Definicija pogleda se piše u jeziku baza podataka koji se zove **SQL**. Korištenje SQL-a pored znanja traži i određenu vještinu.

Ms Access omogućuje definiranje virtualnih/izvedenih relacija pomoću alata Query, na relativno jednostavan način i bez poznavanja SQL-a. Query omogućuje grafičku/tabelarnu konstrukciju pogleda, ali i drugih SQL naredbi/procedura. Access Query je alat koji u velikoj mjeri podržava jezik **QBE (Query By Example)**.

Zanimljivo je da će Query definicija biti generirana SQL jeziku – može se vidjeti ako u skočnom izborniku :

(desna tipka miša) odaberemo SQL View. Nakon SQL View prozora možemo se ponovo vratiti u Query ako odaberemo Design View. Ovo je pogodan način za početno učenje osnova SQL jezika.

Nas u ovoj vježbi zanima njegova upotreba za konstrukciju jednostavnijih pogleda koji su definirani nad jednom relacijom. Query ima tabelarnu strukturu koju popunjavamo dok ne dobijemo potrebnu definiciju:



Pritiskom na desnu tipku u formi Query te izborom „Show Table“, pokazat će se izbornik tablice na temelju koje želimo definirati pogled. Možemo otvoriti više tabela, ali to u ovoj vježbi nećemo raditi.

Nakon toga iz dane tablice prenosimo željene attribute/polja u tabelarnu strukturu definicije.

Redak Sort omogućuje izbor redosljeda prikaza entorki (uzlazno/silazno).

Redak Show omogućuje izbor hoće li se atribut prikazati u pogledu.

Redak uvjet predstavlja uvjet koji će se primijeniti na sve entorke odabrane relacije te će se u pogled uključiti samo one entorke relacije koje uvjet zadovoljavaju.

Kada smo definiciju pogleda završili možemo ga spremati pod određenim imenom te izvesti kad nam zatreba. Pogled će se dinamički izvršavati prema trenutnom sadržaju podatka u relaciji nad kojom je definiran.

### 3.4.5 Validacijski pogledi

Access omogućuje još jednu vrstu ograničenja. Radi se o specijalnom ograničenju kroz validacijski pogled tzv., „Lookup“. Uzmimo npr. tablicu Artickl. Ona sadrži atribut Dobavljač. Kao što je vidljivo atribut Dobavljač je strani ključ u relaciji Artickl, a njegove referencijske vrijednosti moraju egzistirati u relaciji PPartner u atributu IdPP. Zbog toga što između tih dviju relacija postoji veza stranog ključa, atribut Dobavljač može primati samo vrijednosti koje se nalaze tablici PPartner i to u njegovom atributu IdPP. U relacijskoj bazi izbor vrijednosti atributa stranog ključa možemo prikazati kroz validacijski pogled.

To ćemo učiniti na taj način da definiramo validacijski pogled i pridružimo ga formi za dohvat/unos ili ažuriranje atributa stranog ključa.

Najprije ćemo definirati validacijski pogled. Nakon toga možemo taj pogled pridružiti formama gdje je njegova prisutnost potrebna. U Accessu je svaka tablica i svaki pogled u prikazu generička forma – zbog toga se svakoj relaciji može pridružiti validacijski pogled.

Za odabrani atribut relacije/tabele u dijelu „Field Properties“ odaberemo tab „Lookup „(umjesto „General“):

Field Name	Data Type	Description
IdPP	Text	id poslovnog partnera (k
Naziv	Text	naziv poslovnog partner
Mjesto	Text	sjedište u mjestu PP
Adresa	Text	adresa sjedišta PP

Field Properties	
General	Lookup
Display Control	Text Box

Vrijednost za „Display Control“ je „Text Box“ - to znači da atributu nije dodijeljen validacijski pogled. Access omogućuje da se odabere validacijski pogled putem promjene TextBox u ComboBox (kliknite na polje) - pojavit će se tabelarni prikaz za pridruživanje validacijskog pogleda:

Lookup	
Display Control	Combo Box
Row Source Type	Table/Query
Row Source	TipDok
Bound Column	1
Column Count	3
Column Heads	No
Column Widths	2cm;5cm;2cm
List Rows	16
List Width	9cm
Limit To List	Yes
Allow Multiple Values	No
Allow Value List Edits	No
List Items Edit Form	
Show Only Row Source Valu	No

Lookup	
Display Control	Combo Box
Row Source Type	Value List
Row Source	U;- izlazni dokument;I;- izlazni dokument
Bound Column	1
Column Count	2
Column Heads	No
Column Widths	1cm;5cm
List Rows	16
List Width	6cm
Limit To List	Yes
Allow Multiple Values	No
Allow Value List Edits	No
List Items Edit Form	
Show Only Row Source Valu	No

Prikazane su dvije varijante validacije. Lijeva slika prikazuje pogled na bazi validacijskog pogleda, desna na temelju liste vrijednosti. To je određeno tipom „Row source type“. U sam Row Source se specificira naziv validacijskog pogleda ili lista vrijednosti. Ostala svojstva se odnose na format prikaza i mogućnost izbora. Ako postavimo svojstvo „Limit To List“ na „Yes“, forma će dozvoliti unos/promjenu polja ako se vrijednost nalazi u prvom stupcu validacijskog pogleda ili liste vrijednosti. Ako je to svojstvo postavljeno na „No“, moguće vrijednosti nisu ograničene samo na vrijednosti validacijskog pogleda odnosno liste. Svojstvo „Column Count“ označava koliko stupaca validacije će biti prikazano. Ostala svojstva su sama po sebi razumljiva (za detalje pogledajte upute navedene dodatnim materijalima).

## 3.5 Zadaci i sadržaj vježbe

### 3.5.1 Sadržaj vježbe

U ovoj vježbi je potrebno kreirati novu bazu **Vj-3-2013.accdb** koja će sadržavati sve relacijske tablice iz prethodne vježbe proširene s dvije nove tablice.

U bazi je potrebno dodatno kreirati još dvije tablice: Radnik i Dokument. Tablica Radnik sadrži podatke o zaposlenicima. Tablica je referencijski povezana sama sa sobom preko veze „menadžer mu je“ koja određuje koji zaposlenik je menadžer tekućem zaposleniku te na kojoj organizacijskoj razini se zaposlenik nalazi (0 razina označava razinu glavnog menadžera koji nema nadređenog menadžera iznad sebe).

Tablica Dokument sadrži podatke za svaki izdani dokument: poslovnu godinu na koju se odnosi, tip dokumenta prema tablici TipDok, Broj dokumenta pod kojim je izdan, Datum kada je izdan, Šifru poslovnog partnera na koga se dokument odnosi, slobodan tekstualni opis-napomenu uz dokument, Šifru radnika koji je izdao/kreirao dokument te datum i vrijeme upisa dokumenta u bazu.

Tablica **Radnik** sadrži atribute (naziv, tip, opis):

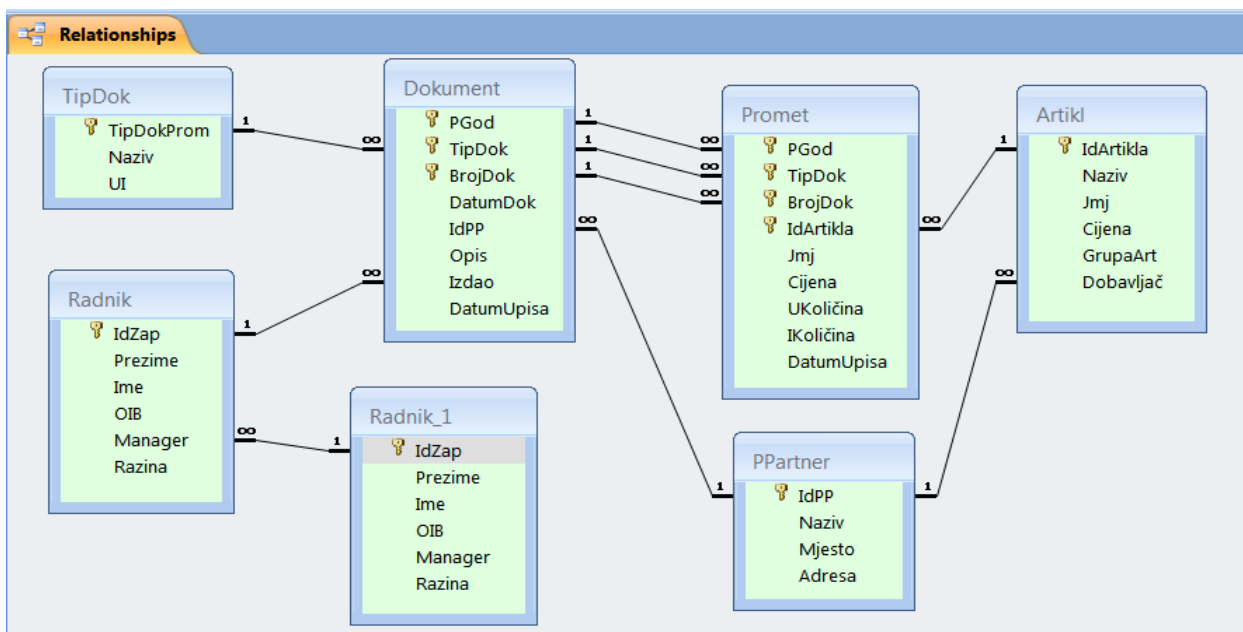
Radnik		
Field Name	Data Type	Description
IdZap	Number	Zaposlenik
Prezime	Text	prezime
Ime	Text	ime
OIB	Text	OIB
Manager	Number	IdZap zaposlenika koji mu je nadređeni manager
Razina	Number	organizacijska razina na kojoj se zaposlenik nalazi (0-najviša,1-niža, 2.....)

Tablica **Dokument** sadrži atribute (naziv, tip, opis):

Dokument		
Field Name	Data Type	Description
PGod	Number	poslovna godina
TipDok	Text	skraćena oznaka za tip/vrsta dokumenta
BrojDok	Number	Broj dokumenta
DatumDok	Date/Time	Datum izdavanja dokumenta
IdPP	Text	id poslovnog partnera (kupca/dobavljača) kome se dokument upućuje ili koji je izdao dok
Opis	Text	napomena-slobodan tekst uz dokument
Izdao	Number	IdZap - (šifra) radnika/skladištara koji je dokument izdao/kreirao/upisao
DatumUpisa	Date/Time	datum i vrijeme kada je dokument upisan u bazu podataka

Za svaku od ovih tablica kreirajte tablicu u bazi s odgovarajućim tipom polja, ograničenjima i primarnim ključem. Potrebno je precizno definirati ograničenja koja nisu definirana u prethodnoj vježbi, zatim definirati validacijske poglede kreirane pomoću Query alata, zatim definirati strane ključeve.

Strane ključeve odredite prema slici:



### 3.5.2 Tijek izvođenja vježbe

- Kreirajte novu bazu Vj-3-2013.accdb na temelju Vj-2-2013.accdb
- U bazu dodajte tablice Radnik i Dokument.
- Provjerite jesu li svi primarni ključevi definirani i jesu li definirani ispravno.
- U tablicama Dokument i Promet atributu Pgod postavite dodatno ograničenje da poslovna godina ne može biti veća od tekuće niti manja od prethodne
- U tablici Radnik atribut OIB ograničite na točno 13 numeričkih znamenaka.
- Za atribut DatumUpisa u tablicama Promet i Dokument postavite defaultnu vrijednost prema trenutnom datumu i vremenu upisa.
- Cijenu u tablicama promet i Artiki ograničite na veću od nule i manju od 100 000.
- Za atribut UI u tablici TipDok postavite validacijsku listu kako je u prethodnom poglavlju navedeno
- Definirajte ograničenja stranog ključa u tablicama Dokument, Radnik, Artiki, Promet prema priloženoj shemi u prethodnom odjeljku.
- Pronađite tablicu u kojoj postoje dva kandidatna ključa
- Pronađite u kojoj tablici je definiran strani ključ nad istom tablicom (tablice roditelj i dijete su iste)



## 3.6 Završna pitanja

Po završetku vježbe, studenti bi trebali bi znati ispravno odgovoriti na sljedeća pitanja (test će se bazirati na ovim pitanjima):

- Što je su to eksplicitna ograničenja atributa?
- Kako se eksplicitna ograničenja atributa definiraju u Accessu?
- Kako se zabrana nul vrijednosti atributa postavlja?
- Što je to default-na (pretpostavljena) vrijednost?
- Kako se validacijsko ograničenje provjerava/testira?
- U kojem slučaju će biti ispisana validacijska poruka?
- Što je nul vrijednost?
- Kakvo nul vrijednost ima značenje u bazi podataka?
- Što je to unique ograničenje, kako se postavlja u Accessu?
- Što je strani ključ?
- U kakvoj su vezi pojmovi strani i primarni ključ?
- Od koliko atributa se može sastojati strani ključ?
- Mora li svaka relacija obavezno imati strani ključ?
- Kako se osigurava referencijalni integritet?
- Koja relacija je u ulozi djeteta, a koja u ulozi roditelja kod referencijalne veze stranog ključa?
- Što su kaskadne operacije/akcije, kako se one izvršavaju?
- Kako se definira/postavlja ograničenje stranog ključa bez kaskadnih akcija?
- U čemu je razlika između nulifikacije i kaskadnog brisanja?
- Što je pogled?
- Što je virtualna relacija?
- Što je izvedena, a što bazna relacija?
- Ako je jedna relacija bazna, a druga iz nje izvedena, u kojem trenutku će se izvedena sinkronizirati s baznom?
- Što je to validacijski pogled? Čemu služi?
- Je li nad jednom baznom relacijom moguće definirati više validacijskih pogleda?
- Što je QBE, a što Access Query, jesu li slični?
- Kako se definira pogled u Accessu?

## 3.7 Zadaci za samostalno produblјivanje znanja

Nakon vježbe diskutirajte s kolegama oko rješenja vježbe. Diskutirajte na temelju argumenata baziranih na uvodu u vježbu te na temelju predavanja iz kolegija baze podataka.

Pronađite na temelju vježbe 1, u NWT bazi kako su definirani pojmovi iz ove vježbe: strani ključevi, referencijalni integritet, upiti (Query), validacijski pogledi,...

Ako ne znate odgovore ili ne razumijete pitanja u prethodnom poglavlju, to je znak da vježbu niste savladali – u tom slučaju uložite dodatni napor da ju samostalno savladate. Pitanja na kraju vježbe su pitanja koja su vezana i za ispit – ne preskačite ih olako.

Provjerite svoje znanje u paru: neka vas kolega/kolegica ispita – nakon toga zamijenite uloge.

## 3.8 Literatura i dodatni materijali

1. <http://office.microsoft.com/hr-hr/access-help>
2. <https://support.office.com/en-za/>

## Vježba 4: Programsko sučelje BP – forme

### 4.1 Motivacija

Podacima u bazi podataka se ne može pristupiti neposredno. Podaci u bazi podataka su pod neposrednim nadzorom sustava za upravljanje bazom podataka (DBMS/SUBP). Korisnici pristupaju bazi putem aplikativnih programa ili odgovarajućih alata kao proširenja DBMS-a. U ovoj ćemo vježbi pomoću Access-ovih razvojnih alata obraditi vrlo skroman uvod u izradu ekranskih formi Accessu. Alat Access Form Designer omogućuje vizualno programiranje ekranskih formi. Detalji se mogu dodatno programirati putem programskog jezika VBA.

Putem ove vježbe studenti će moći steći uvid što u praktičnom smislu predstavlja ekranska forma kao sučelje prema bazi podataka. Kroz ovu vježbu studenti će također, steći osnovna znanja potrebna za samostalno kreiranje formi u Accessu, što će onima koji za to budu imali interesa, omogućiti lakše daljnje učenje.

### Sadržaj vježbe:

<b>PROGRAMSKO SUČELJE BP – FORME</b>	<b>55</b>	
4.1	MOTIVACIJA	55
4.1	CILJEVI VJEŽBE – ISHODI UČENJA	56
4.2	SAMOSTALNA PRIPREMA VJEŽBE	56
4.3	TEORIJSKA PRIPREMA VJEŽBE	57
4.3.1	<i>Semantika u bazi podataka</i>	57
4.3.2	<i>DBMS/SUBP</i>	58
4.3.3	<i>Programsko sučelje prema bazi podataka</i>	58
4.3.4	<i>Ekranske forme</i>	59
4.3.5	<i>Vrste formi</i>	62
4.3.6	<i>Dodavanje podatkovnih objekata</i>	62
4.3.7	<i>Dodavanje ostalih važnijih objekata</i>	63
4.4	ZADACI I SADRŽAJ VJEŽBE	65
4.4.1	<i>Sadržaj vježbe</i>	65
4.4.2	<i>Tijek izvođenja vježbe</i>	65
4.5	ZAVRŠNA PITANJA I ZADACI	66
4.5.1	<i>Pitanja u vezi izložene vježbe</i>	66
4.5.2	<i>Zadaci za samostalno produbljivanje znanja</i>	66
4.6	LITERATURA I DODATNI MATERIJALI	66

## 4.2 Ciljevi vježbe – ishodi učenja

Od studenata se očekuje da, nakon uspješno završene vježbe:

- Shvate pojam značaj aplikacijskih programa kao sučelja prema bazi podataka
- Nauče izraditi jednostavne podatkovne forme u Accessu
- Razumiju kako se podaci iz baze povezuju s formama i objektima na formi
- Shvate ulogu podatkovnih pogleda i njihovo korištenje na formama

## 4.3 Samostalna priprema vježbe

Prije dolaska u lab, obaveze studenata su:

- Proučite prethodnu materiju sa predavanja iz kolegija BP
- Završite prethodnu vježbu ako ju niste izradili ili razumjeli do kraja
- Proučite pripremne materijale za ovu vježbu
- Utvrdite prethodnu vježbu – pitanja u prethodnoj vježbi će predstavljati temelj za blic-test u ovoj vježbi
- Minimalni skup znanja potrebnih za izvođenje vježbe:
  - Što je semantika podataka i gdje treba biti ugrađena?
  - Što je to programsko sučelje?
  - Forme kao programsko sučelje?
  - Vrste formi?
  - Vrste objekata na formi i njihova namjena
  - Svojstva objekata na formi – grupe svojstava i njihovo značenje
  - Najvažnija svojstva objekata na formi

## 4.4 Teorijska priprema vježbe

U pripremi za ovu vježbu dobro proučite sljedeće materijale.

### 4.4.1 Semantika u bazi podataka

Baza podataka sadrži dvije vrste podataka: podatke u užem smislu i metapodatke (opise podataka). Podaci u užem smislu su podaci zbog kojih je baza podataka izgrađena. Metapodaci predstavljaju podatke koji semantički opisuju podatke. Bez semantičkog opisa, značenje podataka je nepoznato. Semantički opis podataka može biti u bazi podataka ili u aplikacijskim programima koji koriste podatke iz baze podataka. Aplikacijski programi predstavljaju sučelje pomoću kojega korisnici pristupaju bazi podataka radi čitanja, pretraživanja, dodavanja ažuriranja i brisanja podataka u bazi.

Semantiku u relacijskoj bazi podataka možemo ugraditi u tri oblika:

- **strukturalna semantika** – proizlazi z strukture objekata odnosno strukturalnih ograničenja,
- **integritetska semantika** - kroz specifikaciju dodatnih pravila integritetskih ograničenja,
- **proceduralna semantika** – prikazana je kroz proceduralne akcije nad podacima.

Semantiku ugrađujemo u bazu podataka modeliranje podataka za određeno područje interesa (problem). Dobiveni model podataka će se implementirati (ugraditi) u shemu baze podataka. Ako takav model ne sadrži dovoljnu ili ispravnu semantiku (tj. ako nije u skladu s zahtjevima područja interesa), onda niti shema baze podataka neće sadržavati potrebnu semantiku. Često se umjesto pojma semantika upotrebljava pojam poslovna logika (**business logics**). Radi se o poslovnim pravilima (**business rules**) u modeliranom sustavu odnosno u području interesa i potrebi da se ona obuhvate modelom podataka i ugrade u shemu baze podataka. Pritom se često naglašava razlika semantike oblika i načina prikaza podataka, što se onda označava kao prezentacijska semantika ili prezentacijska logika (**presentation logics**)

Koncept baze podataka prirodno zahtijeva da semantika podataka bude zapisana u bazi. U praksi se to nikada neće dogoditi u potpunosti. Dva su osnovna razloga za to. Jedan proizlazi iz neznanja i/ili nemara konstruktora baze podataka, a drugi iz nesavršenosti DBMS-a u koji se baza implementira.

Ako konstruktor baze podataka nije dovoljno dobro konstruirao bazu, ispustivši dio semantike iz baze, onda ta semantika mora biti ugrađena u aplikacijske programe koji koriste podatke iz baze. Kod relacijskih baza podatka omogućeno je da semantika bude u velikoj mjeri ugrađena u samu bazu. Međutim, dio semantike praktično nikada neće biti moguće ugraditi u bazu. Osnovni razlog nije u nemaru i li neznanju konstruktora baze, nego u nesavršenosti inherentnog modela podataka na kome je

konkretni DBMS izgrađen, ali isto tako iz zbog nepotpune implementacije relacijskog modela podataka. Zbog toga, čak i kod pažljivog modeliranja i konstrukcije baze podataka, dio semantike će se morati ugraditi u aplikacijske programe. To je nužno zlo, ali upravo zbog toga treba obratiti posebnu pažnju profesionalnom modeliranju i konstrukciji sheme baze podataka. Problem s eksterno ugrađenom semantikom je u tome što ju je s vremenom razvoja aplikativnih programa sve teže kontrolirati i održavati konzistentnom. Interno ugrađenu semantiku (ugrađenu u BP), je moguće centralno efikasno kontrolirati i održavati konzistentnom bez obzira na promjene koje su za bazu podatka karakteristične. Pritom treba naglasiti da se ne mijenjaju samo podaci nego i metapodaci, pa to onda dodatno usložnjava problem konzistencije.

U prethodnim vježbama smo objasnili na primjeru Ms Accessa kako možemo u bazu podataka ugraditi strukturalnu i dio integritetske semantike. U idućim vježbama ćemo detaljnije proučiti dodatnu integritetsku semantiku, a osobito ćemo se baviti implementacijom proceduralne semantike kroz rad sa SQL-jezikom.

#### 4.4.2 DBMS/SUBP

Podaci i metapodaci su pod stalnim nadzorom DBMS-a. **DBMS** dolazi kao akronim od engleskog naziva: Database Management System, što se može prevesti kao: **SUBP**-sustav za upravljanje bazom podatka. DBMS izolira podatke u bazi podataka od okoline. Aplikacijski programi ne mogu direktno pristupiti podacima u bazi. Oni pristup podacima ostvaruju posredno preko DBMS-a izdajući mu zahtjeve za podacima u obliku komandi. Na taj način DBMS osigurava integritet podataka i unificiranu manipulaciju podacima. DBMS te zahtjeve interpretira izvršavajući traženu manipulaciju nad podacima. Ti zahtjevi mogu biti pretraživanje/čitanje, ubacivanje/dodavanje, ažuriranje /promjena te brisanje/uklanjanje podataka iz baze.

#### 4.4.3 Programsko sučelje prema bazi podataka

Pristup podacima se ostvaruje putem aplikacijskih programa. Aplikacijski programi mogu biti (najčešće jesu) potpuno neovisni o DBMS-u, ili mogu biti dodatni alati uz DBMS. Aplikacijski programi mogu biti izrađeni pomoću alata temeljenih na nekom od viših programskih ili skriptnih jezika. Rezultat su aplikativni programi koji se izvršavaju kao klasične desktop aplikacije ili pak kao web-aplikacije. Kod klasičnih aplikacija prezentacijska razina je „desktop“. Web aplikacije se izvode djelomično na web-serveru a djelomično na klijentskom internet pretraživaču (browseri : IE, FF, Chrome, Opera,..). Prezentacijska razina web aplikacija koje koriste bazu podataka je internet pretraživač, a baza se nalazi na udaljenom serveru.

Ms Access sadrži integrirani skup alata kao razvojnu okolinu za izradu aplikacijskih programa. Access omogućuje klijent server rad uz desktop aplikacije, ali također i web pristup bazi putem Microsoft Sharepoint servisa.

Kod klasičnog „desktop“ pristupa Accessove aplikacije se zasnivaju na sljedećim alatima:

- Form designeru –alatu za vizualnu izradu ekranskih formi,
- Visual Basic for Applications – programskom jeziku zasnovanom na Visual Basicu s editorom kompajlerom i debagerom,
- Report designeru - alatu za vizualno oblikovanje izvještaja pogodnih za ispis ali i za neposredni pregled na ekranu,
- Macro kreatoru – alatu za tabelarno programiranje za ne programere.

Prezentacijska razina podataka iz baze može biti u obliku:

- Ekranskih formi,
- Izvještaja.

U ovoj vježbi ćemo u vrlo skraćenom obliku izložiti koncepte ekranskih formi u Accessu. U nekim drugim aplikacijskim razvojnim alatima koncepti formi mogu biti donekle različiti, ali ipak u velikoj mjeri slični izloženima.

#### 4.4.4 Ekranske forme

Ekranske forme su temeljni oblik dinamičkog sučelja prema podacima u bazi podataka. Jedna aplikacija (aplikacijski program) se sastoji od jedne ili više formi. Svaka forma predstavlja neku funkcionalnu cjelinu. Prikaz odnosno izvršenje forme se izvodi putem pozivanja forme s neke druge forme ili iz trake izbornika. Formi može, ali ne mora biti pridružen izvor podataka. Izvor podataka (**Record Source**) može biti ili neka bazna tablica ili neka izvedena tablica (pogled, query). Forma je načelno povezana s cijelim izvorom podataka (sa svim entorkama), ali u svakom trenutku se zna koja entorka je tekuća (Current Row). To je tzv. koncept kursora koji pokazuje na tekuću entorku. Pritom postoji mogućnost kretanja na sljedeću i prethodnu entorku. Iako u relacijskom modelu podataka nema svojstva koja je entorka u relaciji prva, a koja sljedeća ili prethodna, formi pridružene tablice su na određen način sortirane (na to se može utjecati kroz svojstva forme, ali i programski). Pritiskom na određene tipke tastature (PgUp, PgDn, Home, End,..), dugmadi na formi (navigacijske tipke) ali programskim komandama, moguće je mijenjati koja će entorka biti tekuća. Promjena tekuće entorke će automatski izazvati promjenu vrijednosti povezanih podataka na formi.





Property Sheet		Property Sheet	
Selection type: Form		Selection type: Form	
Form		Form	
Format	Data	Event	Other
On Current			
On Load			
On Click			
After Update			
Before Update			
Before Insert			
After Insert			
Before Del Confirm			
On Delete			
After Del Confirm			
On Dirty			
On Got Focus			
On Lost Focus			
On Dbl Click			
On Mouse Down			
On Mouse Up			

Property Sheet	
Selection type: Form	
Form	
Format	Data
Pop Up	No
Modal	No
Display on SharePoint Site	Follow Table Setting
Cycle	All Records
Ribbon Name	
Toolbar	
Shortcut Menu	Yes
Menu Bar	
Shortcut Menu Bar	
Help File	
Help Context Id	0
Has Module	No
Use Default Paper Size	No
Fast Laser Printing	Yes
Tag	

Napomena: Različiti tipovi objekata-kontrola će imati različiti skup mogućih svojstava u pojedinoj kategoriji. To proizlazi iz smisla namjene i funkcionalnosti pojedinog tipa kontrole.

Programer može direktno u tabeli svojstava ili putem programiranja promijeniti vrijednost svojstva u okviru mogućih vrijednosti. Vrijednost svojstva je ispisana u desnom stupcu tabele svojstava. Vrijednosti svojstava mogu biti u jednom od traženih oblika kao što su: tekst, broj, naziv funkcije/procedure, Yes/No.

Ako želite dobiti detaljne upute o značenju pojedinog svojstva, potrebno je:

- Pozicionirati se mišem na to svojstvo u tabeli svojstava,
- Pritisnuti tipku F1.

Svojstva forme i ostalih kontrola su grupirana u četiri grupe:

**Format** – svojstva koja određuju izgled forme/kontrole,

**Data** – svojstva koja se odnose na to kako i koji su podaci (tablice, pogledi) vezani uz formu ili kontrolu,

**Event** – opisuju kako će forma/kontrola reagirati na neke događaje na koje može reagirati,

**Other** – ostala svojstva forme/kontrole.

**Format** - skup svojstava određuje kako će se kontrola zvati, dimenzije, poziciju, boju oblik, svojstvene kontrole,...

**Data** - skup svojstava određuje način na koji je forma povezana sa podacima odnosno koje polje iz formi pridruženog izvora podataka će biti pridruženo kontroli. Zatim definira je li prikazane podatke moguće ispravljati ili su samo za prikaz.

**Event** - skup svojstava nabraja sve događaje na koje forma/kontrola može reagirati. Događaji mogu biti npr., postavljanje kontrole u fokus, klik, dvostruki klik, promjena stanja/ažuriranje, napuštanje kontrole,...

Reakcija predstavlja izvršenje određene programske rutine (funkcije ili procedure) napisane kao makro program ili program u VBA. Ako se želi da kontrola/forma reagira na događaj, onda se događaju pridruži naziv rutine koji treba izvršiti kad se događaj dogodi. Svojstva forme i kontrola () na njoj se dobivaju pritiskom desne tipke miša, ali prije toga dotični objekt mora biti u fokusu (selektiran kao tekući).

#### 4.4.5 Vrste formi

S obzirom na prikaz povezanih podataka iz izvora podataka, forme mogu biti:

- **Single form** – to su jednostruke forme sa slobodno uređenim kontrolama (to je najčešći oblik forme).
- **Continuous form** – to je oblik forme u kome se jednostruka forma multiplicira ispod prethodne za svaku entorku pridruženog izvora podataka.
- **Datasheet form** – to je tabelarni oblik forme koji prikazuje izvor podataka u tabelarnom obliku.
- **Split form** – to je kombinacija single form i datasheet form.
- **Pivot table** – omogućuje prikaz dinamičke datasheet forme gdje se polja mogu prenositi i mijenjati pozicije
- **Pivot Chart** je forma koja omogućuje grafički prikaz podataka i to u obliku dinamičkog grafikona.

Tip forme s obzirom vrstu prikaza podataka se određuje kroz svojstvo: **Format->Default View**

S obzirom na to kako se forma ponaša u odnosu na druge forme određuju postavke dva svojstva iz kategorije **Other**: „Pop Up“ i „Modal“:

**Pop Up** svojstvo određuje hoće li forma prilikom otvaranja iskočiti ispred drugih formi (=Yes) ili će se ponašati normalno (=No).

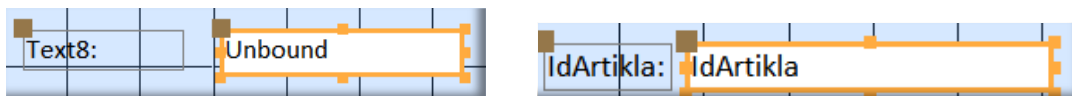
**Modal** svojstvo određuje hoće li se forma kada dođe u fokus tj. kad se otvori, moći napustiti, a da se prije toga nije zatvorila (=No), ili će biti onemogućen prelazak na druge istovremeno otvorene forme, prije nego što se forma zatvori (=Yes).

#### 4.4.6 Dodavanje podatkovnih objekata

Neki objekti kontrole mogu biti povezani s podacima odnosno izvorima podataka. Najvažniji među njima je „Text Box“. Tekst-polje im dva dijela: Labelu- koja opisuje polje i samo polje koje sadrži podatak.

Za vrijeme dok se forma nalazi u Design modu, u polju će biti ispisan naziv podatka/polja iz izvora, a ako tekst boks nije povezan s nekom podatkom pisat će „Unbound“. U labeli će pisati naziv labela – moguće ga je promijeniti prema potrebi u design modu. Kad se forma otvori /izvede, u tekst polju će pisati sadržaj podatka odnosno tekst polja. Labela će ostati nepromijenjena (iako ju je moguće programski mijenjati u svakom trenutku)

Primjer tkst polja (Text Box) – lijevo nepovezanog (unbound), i desno povezanog s izvorom podatka (Bound) odnosno povezanog s poljem „IdArtikla“:

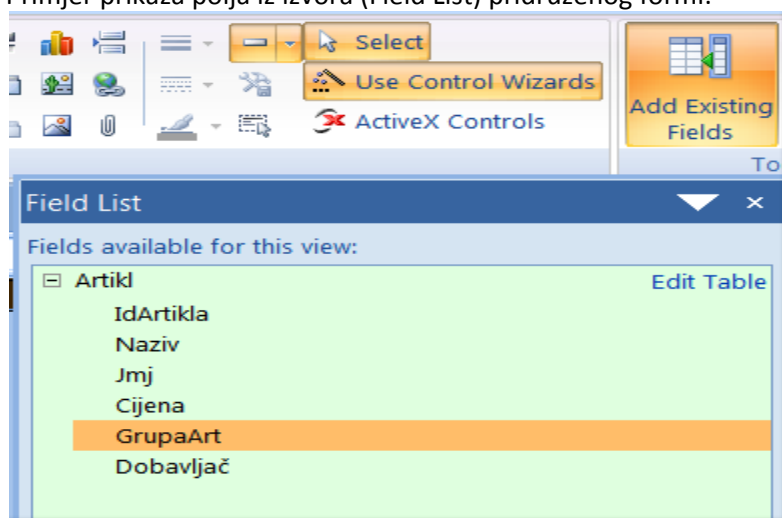


Objekti/kontrole na formi mogu biti povezani s izvorom podataka ili mogu biti nepovezani. Povezanost objekta s izvorom podataka je najčešće preko onog izvora podataka koji je pridružen formi.

Kada je formi pridružen izvor podataka, tekst-polja se mogu na formu dodavati na način da su direktno povezana s izvorom podataka odnosno određenim poljem izvora. Da bi dodali neko polje s izvora podataka na formu moramo učiniti sljedeće:

- U izborniku alatne trake izaberemo **Design->Add Existing Fields**
- Prikazat će se tabela polja u izvoru podataka koji je pridružen formi,
- Odaberemo polje koje želimo dodati na formu i mišem ga prenesemo na formu,
- Pozicioniramo tekst-polje na formi, uredimo ga i prilagodimo labelu (sve kroz njihov skup svojstava)

Primjer prikaza polja iz izvora (Field List) pridruženog formi:



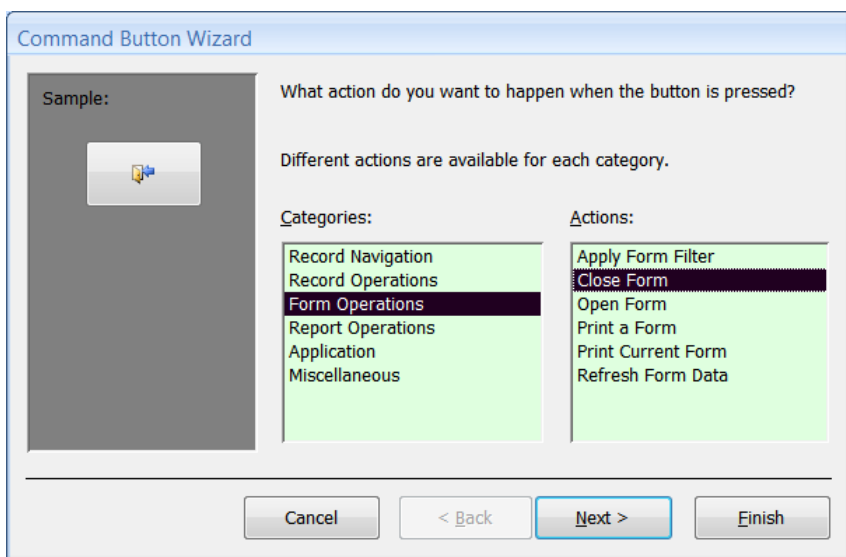
#### 4.4.7 Dodavanje ostalih važnijih objekata

Navigacijske tipke su objekt na podatkovnoj formi koje omogućuju navigaciju (kretanje) po entorkama (redovima podataka) pridruženog izvora podataka. Navigacijske tipke će se naći u lijevom donjem kutu forme:



Navigacijske tipke aktiviramo svojstvom forme: **Format->Navigation Buttons**.

Kako bi omogućili pokretanje potrebnih akcija na formi koje bi trebale biti navedene i izvršene eksplicitno, koristimo objekt „Dugme“ (Button). U izborniku odaberemo Create->Button te zatim nacrtamo oblik dugmeta na formi. Access će prikazati dijaloški prozor u kome trebamo odabrati akciju koja će biti izvršena pritiskom na dugme kad forma bude aktivna. Npr., možemo odabrati da se tekuća forma zatvori ili da se otvori nova forma:



U ovom primjeru možemo vidjeti da će nas Access voditi kroz proceduru automatskog kreiranja programskog koda za ponuđenu akciju, te će se taj kod automatski povezati s formom (ništa ne moramo programirati). Nakon toga dobiveni kod možemo urediti prema dodatnim zahtjevima ako ih bude.

## 4.5 Zadaci i sadržaj vježbe

### 4.5.1 Sadržaj vježbe

- U ovoj vježbi je potrebno preuzeti ispravnu bazu Vj-3-2013.accdb, popuniti je podacima (desetak redova odnosno entorki po tablici) i preimenovati u VJ-4-2013.accdb.
- Nakon toga treba izraditi forme koje će omogućiti prikaz podataka i to za svaku tablicu.

### 4.5.2 Tijek izvođenja vježbe

- Kreirajte novu bazu Vj-4-2013.accdb prema ispravnoj bazi iz prošle vježbe
- Popunite bazu smislenim podacima (ili preuzmite popunjenu).
- Za svaku tablicu izradite po jednu formu.
- Isprobajte mijenjati tip prikaza podataka za izrađene forme (Datasheet, Single Form,..)
- Izradite jednu formu „Glavna forma“ na kojoj će te postaviti dugmad pomoću kojih će te moći otvoriti svaku od prethodnih formi.
- Na glavnoj formi dodajte dugme kojim ćete zatvoriti aplikaciju.
- Eksperimentirajte promjenom svojstava pojedinih formi.
- Umjesto tablica formama pridružite poglede iz prethodne vježbe.

## 4.6 Završna pitanja i zadaci

### 4.6.1 Pitanja u vezi izložene vježbe

Po završetku vježbe, studenti bi trebali znati ispravno odgovoriti na sljedeća pitanja (test će se temeljiti na ovim pitanjima):

- Što je to sučelje prema BP?
- Zbog čega se bazi ne može pristupiti neposredno?
- Što je posrednik između aplikacijskih programa i podataka u BP?
- Što su aplikacijski programi i kako se povezuju s BP?
- Kave vrste prezentacijskih razina podataka poznajete?
- Što je to ekranska forma?
- Kako se podacima u bazi može pristupiti ?
- Kako se forma povezuje s podacima?
- Koje vrste forme su Accessu moguće?
- Što su to svojstva forme?
- Koje kategorije svojstava postoje i što znače?
- Što su to objekti forme?
- Što su događaji i kako forma na njih reagira?
- Što znači pojam izvor podataka?
- Kava je uloga izvora podataka na formi i njenim objektima?
- Što je navigacija na formi?
- Što je tekst-polje, kakav je to objekt?
- Što je dugme, kakav je to objekt na formi, čemu služi?
- Kako se forme mogu međusobno povezati?

### 4.6.2 Zadaci za samostalno produblivanje znanja

Nakon vježbe diskutirajte s kolegama oko rješenja vježbe. Raspravljajte na temelju argumenata temeljenih na uvodu u vježbu te na temelju predavanja iz kolegija baze podataka.

Ako ne znate odgovore ili ne razumijete pitanja u prethodnom poglavlju, to je znak da vježbu niste savladali – u tom slučaju je potreban dodatni napor da ju samostalno savladate. Pitanja na kraju vježbe su pitanja koja su vezana i za ispit – ne preskačite ih olako.

Provjerite svoje znanje u paru: neka vas kolega/kolegica ispita – nakon toga zamijenite uloge.

## 4.7 Literatura i dodatni materijali

- 1) [http://ecdl.hgk.hr/Pdf/W7\\_2010/ITDesk\\_Prirucnik\\_5\\_baze\\_podataka\\_microsoft\\_access\\_2010.pdf](http://ecdl.hgk.hr/Pdf/W7_2010/ITDesk_Prirucnik_5_baze_podataka_microsoft_access_2010.pdf)
- 2) [http://www.itdesk.info/prirucnik\\_baze\\_podataka\\_microsoft\\_access\\_2010.pdf](http://www.itdesk.info/prirucnik_baze_podataka_microsoft_access_2010.pdf)
- 3) <https://support.office.com/en-au/article/Create-a-form-that-contains-a-subform-264731d6-ca69-4204-94d8-c266fe084102>

**Vježba 5:****Programsko sučelje BP  
– subforme i izvještaji |****5.1 Motivacija**

U ovoj ćemo se vježbi nakon proširivanja znanja o ekranskim formama, te se fokusirati na izvještaje kao jedan od standardnih prikaza podataka iz baze podataka. Access raspolaže s vrlo pristupačnim alatima za kreiranje subformi i izvještaja. Subforme najčešće odražavaju/prikazuju podatkovni odnos roditelj/djeca (master/child), kao što je npr. odnos dokument/stavke dokumenta. Takvi odnosi su česti u praksi pa je njihovo praktično razumijevanje važno. Osobito je važno pritom, uočiti konzistenciju između podatkovnog i aplikativnog modela. Izvještaji su neizostavni dio gotovo svake aplikacije. U Accessu su izvještaji čvrsto integrirani u razvojni koncept te se vrlo dobro uklapaju u aplikacijski model. Kao i kod prethodne vježbe, glavni cilj ove vježbe nije naučiti razvijati aplikacije u Accessu, nego pokazati na jednostavnom praktičnom primjeru povezanost formi i izvještaja s podatkovnim modelom relacijske baze podataka. No, i pored toga, aplikativno iskustvo stečeno kroz vježbe može studentima biti korisno, ali i stimulativno za vlastito učenje ne samo Accessa.

**Sadržaj vježbe:****PROGRAMSKO SUČELJE BP – SUBFORME I IZVJEŠTAJI | 67**

5.1	MOTIVACIJA	67
5.2	CILJEVI VJEŽBE – ISHODI UČENJA	68
5.3	SAMOSTALNA PRIPREMA VJEŽBE	68
5.4	TEORIJSKA PRIPREMA VJEŽBE	69
5.4.1	<i>Vrste povezivanja podataka s objektima forme</i>	69
5.4.2	<i>Subforme</i>	70
5.4.3	<i>Izvještaji</i>	72
5.4.4	<i>Struktura izvještaja</i>	73
5.4.5	<i>Kreiranje izvještaja</i>	74
5.5	ZADACI ZA STUDENTE TIJEKOM VJEŽBE	78
5.5.1	<i>Sadržaj vježbe</i>	78
5.5.2	<i>Tijek izvođenja vježbe</i>	78
5.6	ZAKLJUČNA PITANJA I ZADACI	79
5.6.1	<i>Pitanja u vezi izložene vježbe</i>	79
5.6.2	<i>Zadaci za samostalno produbljivanje znanja</i>	79
5.7	LITERATURA I DODATNI MATERIJALI	80

## 5.2 Ciljevi vježbe – ishodi učenja

Od studenata se očekuje da, nakon uspješno završene vježbe:

- Shvate pojam subforme.
- Nauče definirati subformu i povezati ju s formom na kojoj se nalazi.
- Shvate pojam izvještaja i njegovog povezivanja s formom.
- Nauče izraditi jednostavne izvještaje u Accessu .

## 5.3 Samostalna priprema vježbe

Prije dolaska u lab, obaveze studenata su:

- Proučite prethodnu materiju sa predavanja iz kolegija BP
- Završite prethodnu vježbu ako ju niste izradili ili razumjeli do kraja
- Proučite pripremne materijale za ovu vježbu
- Utvrdite prethodnu vježbu – pitanja u prethodnoj vježbi će predstavljati temelj za blic-test u ovoj vježbi
- Minimalno potrebna znanja za pristup vježbi:
  - Što je ekranska forma
  - Koji su tipični objekti na formi
  - Kako se povezuje forma s podacima u relaciji/pogledu
  - Kako se povezuju objekti na formi i polja s izvora podataka tebllice/pogleda
  - Što je izvještaj, po čemu se razlikuje od ekranske forme
  - Što znači hijerarhijska forganizacija forme
  - Kako se forma povezuje s podacima u bazi podataka



## 5.4 Teorijska priprema vježbe

U pripremi za ovu vježbu dobro proučite sljedeće materijale.

### 5.4.1 Vrste povezivanja podataka s objektima forme

U prethodnoj smo vježbi objasnili da formi može, ali i ne mora, biti pridružen izvor podataka u obliku relacijske tablice ili pogleda. Prisjetimo se, ako je formi pridružen izvor podataka, forma se u svakom trenutku nalazi na određenom redu (entorki) pridruženog izvora podataka. Dakle, postoji svojstvo (tzv. kursor) koji pokazuje na trenutni aktualni redak (Current Row). Kad se forma otvori currentrow će pokazivati na prvi redak izvora (ako programski nije postavljeno da to bude neki drugi red). Redoslijed prikaza će biti određen preme sortnom redoslijedu koji je izvoru ili formi pridružen. Ako nije pridružen redoslijed, onda će DBMS prikazivati podatke u redoslijedu prema vlastitoj logici (obično je to redoslijed kojim su entorke dodavane u tablicu).

Kretanje po izvoru podataka se može zamisliti kao kretanje po redovima tabele: gore(prethodni), dolje(sljedeći), prvi redak (vrh tabele), zadnji redak (dno tabele). Kretanje se izvodi pomoću navigacijskih tipki ili pomoću tipki tastature: PgUp, PgDown, Home, End.

Objekti na formi (kontrola) isto tako mogu imati pridružene izvore podataka. Nekim objektima poput dugmadi (Button) i labela, nema smisla pridruživati izvore podataka, što proizlazi iz njihove temeljne namjene odnosno karakteristike. Postoje tri vrste izvora podataka koje možemo pridružiti objektima (kontrolama) na formi:

**Izvor skupa podataka – tabele** (naziva se: **Record Source**) koji omogućuju povezivanje skupa entorki (iz određene bazne ili virtualne relacije). Ovakav oblik izvora podataka se pridružuje formi.

**Izvor pojedinačnog podatka** (naziva se: **Control Source**). Ovakav oblik izvora podataka se pridružuje objektima na formi koji sadrže jedan pojedinačni podatak. Najčešći je to slučaj s objektom tekst-polje, ali to vrijedi i za ostale objekte koji mogu biti povezani s pojedinačnim podatkom. Ako je to podatak iz skupa atributa izvora koji su pridruženi formi, onda će se njegova vrijednost automatski mijenjati u zavisnosti od promjene tekuće entorke (odnosno kretanjem kroz izvor skupa podataka).

**Izvorni objekt** (naziva se **Source Object**) koji omogućuje da se objektu subforma pridruži izvor podataka preko nekog drugog objekta (forme, izvještaja, tabele, pogleda). Npr., subformi se pridružuje druga forma odnosno njoj pridružen izvor skupa podataka.

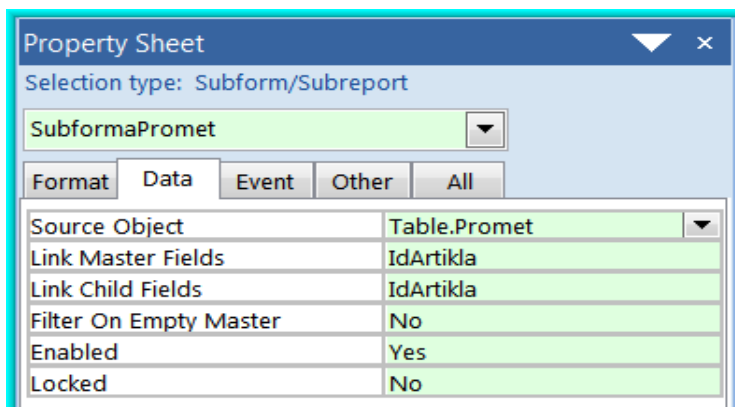
## 5.4.2 Subforme

Na formu se može postaviti jedna ili više subformi. Subforma je objekt-kontrola koji u okviru dane forme u pravokutnom prozoru prikazuje neku drugu formu koja je toj subformi pridružena. Ta forma može biti neka forma koju smo prije kreirali ili generička forma za tabelu ili pogled. Budući da Access svakoj relacijskoj tablici, a i svakom pogledu, pridružuje defaultni tabelarni prikaz (generičku tabelarnu formu), onda postavljanjem objekta-izvora za subformu (**Source Object**) pridružujemo zapravo generičku formu.

Specifičnost subforme je u tome da može prikazati tabelarni uvid u podatke koji su na neki način definirani /povezani s nekim podacima na formi.

Npr., na formi Artikal možemo staviti subformu Promet koja će prikazivati samo promet istog artikla – prema tekućoj vrijednosti artikla na glavnoj formi. Svaki puta kad se ta vrijednost promijeni (bilo da se krećemo po entorkama glavne forme bilo da upišemo novu vrijednost u polje IdArtikla), promijenit će se sadržaj subforme tako da prikazuje promet za promijenjeni artikal. To će se događati ako smo definirali filter koji se sastoji od dva dijela:

- **LinkMasterFields** – navest ćemo popis polja iz glavne forme koja će odgovarati poljima iz subforme (razdvojeno točka zarezom: ;)
- **LinkChildFields** – navest ćemo popis polja iz subforme tako da odgovaraju vrijednostima iz glavne forme.



Ostala svojstva subforme imaju sljedeće značenje:

- **FilterOnEmptyMaster** – u slučaju da su polja na glavnoj formi prazna može se pojaviti greška u povezivanju – postavimo li ovu vrijednost na NO , to se neće dogoditi.
- **Enabled** – ima značenje kao i kod drugih kontrola – subforma neće biti aktivna.
- **Locked** – ako je postavljeno na No, podaci se u subformi neće moći mijenjati, a ako je postavljeno na Yes, podatke ćemo moći uređivati.

Subformu ćemo kreirati tako da kliknemo ikonu subforme u izborniku Design, a zatim na formi obilježimo poziciju subforme u obliku pravokutnika. Nakon toga će se otvoriti Wizard-alat koji nas vodi kroz postupak definiranja subforme:

You can use an existing form to create your subform or subreport, or create your own using tables and/or queries.

What data would you like to use for your subform or subreport?

Use existing Tables and Queries

Use an existing form

Artikl subform

SubForm Wizard

Which fields would you like to include on the subform or subreport?

You can choose fields from more than one table and/or query.

Tables/Queries

Table: Artikl

Available Fields:

GrupaArt
Dobavljač

Selected Fields:

IdArtikla
Naziv
Jmj
Cijena

Buttons: >, >>, <, <<

Buttons: Cancel, < Back, Next >, Finish

SubForm Wizard

Would you like to define which fields link your main form to this subform yourself, or choose from the list below?

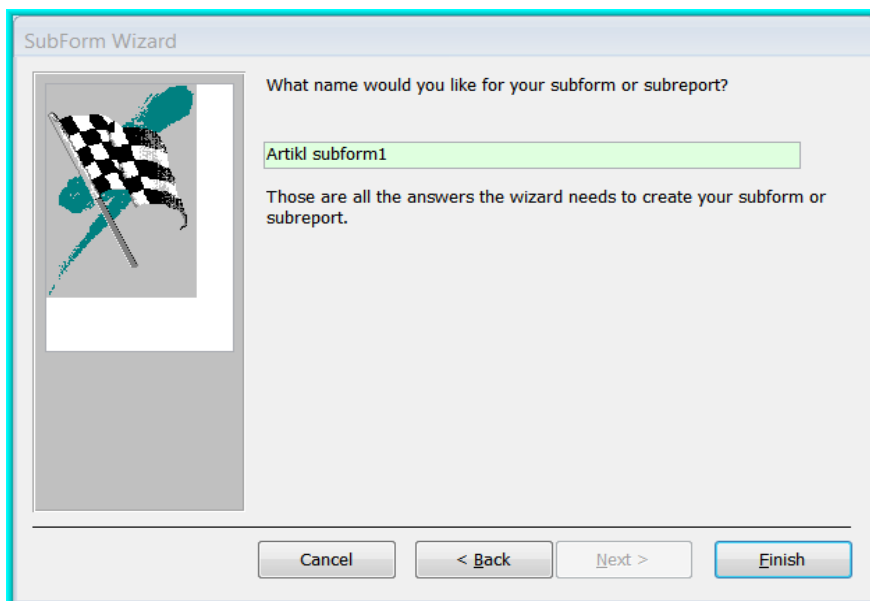
Choose from a list.  Define my own.

Form/report fields: Subform/subreport fields:

IdArtikla IdArtikla

Show Artikl for each record in Artikl using IdArtikla

Buttons: Cancel, < Back, Next >, Finish



Ostaje nam mogućnost da subformu definiramo kroz njena svojstva tako da prekinemo Subform Wizard (čarobnjak subforme) ili da nastavimo definiciju putem čarobnjaka forme. Ako nastavljamo pomoću čarobnjaka, prvo što moramo je izabrati hoće li objekt-izvor za subformu biti tablica/pogled ili će to biti neka postojeća forma. Ako smo odabrali tablicu/pogled (Query), treba uključiti sva ona polja (attribute) koje želimo prikazati na subformi. Nakon toga treba definirati filter između podataka glavne forme i subforme tj., kako će subforma biti povezana s formom – uključit ćemo polja s forme i odgovarajuća polja sa subforme (to će djelovati kao filter podataka subforme prema vrijednosti podataka na formi (onih koje smo uključili u filter/vezu).

Na kraju kliknemo Finish a čarobnjak će generirati novu formu (možemo joj promijeniti naziv) koja će biti na glavnoj formi prikazana kao subforma.

Ako subformu nismo definirali pomoću čarobnjaka subforme, za nju moramo ručno definirati barem objekt-izvor (source objekt), Filter nije nužan, tj., ako Link svojstva ostanu prazna, neće djelovati filtriranje podataka subforme. Source object, Filter i ostala svojstva je moguće programski definirati dinamički unutar programske logike.

### 5.4.3 Izvještaji

**Izvještaji (Reports)** predstavljaju statički prikaz podataka iz baze, pogodan za ispis na papiru. Izvještaji mogu funkcionirati i kao subforme, ali njihova je osnovna namjena strukturirani ispis podataka.

Forme omogućuju dinamički i statički prikaz podataka. Forme se mogu ispisati na pisač, ali one nisu za to prilagođene - njihova je osnovna namjena ekranski prikaz i interakcija korisnika s bazom podataka. Ako se forme i izvještaji međusobno kombiniraju, može se izvještajima dati potrebna dinamičnost. To se postiže na taj način da se preko forme filtrira/odabire skup podataka koji će se

ispisati na izvještaju, ali i svojstva izgleda izvještaja. Složeniju dinamičnost izvještaja je moguće postići uz dodatno programiranje. Svakoj formi i izvještaju mogu biti pridružene programske rutine (moduli, funkcije, procedure, strukture podataka). Te programske rutine će se pridružiti određenim događajima (nalaze se u Event kategoriji svojstava) forme odnosno izvještaja te će se automatski pokrenuti /izvesti kad se događaj dogodi.

Izvještaju moramo pridružiti određeni izvor skupa podataka (Record Source). Izvor skupa podataka može biti bazna relacija ili pogled (Query). Za razliku od izvještaja, formi možemo, ali ne moramo pridružiti izvor skupa podataka. Npr., formi izborniku u trećoj vježbi nije bilo potrebno pridružiti izvor podataka.

Ipak, moguće su rijetke situacije kad nije potrebno niti izvještaju pridružiti izvor podataka. Npr., ako želimo izvještaj koji prikazuje neki fiksni sadržaj/tekst koji je potrebno moći ispisati, onda mu nije potrebno pridružiti izvor podataka.

#### 5.4.4 Struktura izvještaja

Koncept izvještaja u Accessu ima hijerarhijsku strukturu na čijoj najnižoj razini se nalaze podaci čiji se ispis ponavlja u obliku jednostavnog ili složenog reda na izvještaju. Taj dio izvještaja se zove detalji (Detail). Podaci prikazani u detaljima se mogu grupirati na jednu ili više razina. Grupiranje ima za posljedicu stvaranje zaglavlja (Header) i/ili podnožja (Footer) grupe. Vrlo je važno na koji način su podaci u izvoru (Record source) sortirani te na koji način su sortirani u izvještaju – od toga zavisi smislenost grupiranja i redoslijed ispisa podataka na izvještaju.

Hijerarhijska struktura od n razina grupiranja izvještaja obuhvaća sljedeće koncepte (objekte izvještaja):

**Report Header** - zaglavlje izvještaja ; pojavljuje se na početku izvještaja

**Page Header** - zaglavlje stranice; ispisat će se na početku svake stranice izvještaja

**Xn Header** - zaglavlje grupe najviše razine (X označava polje grupiranja, a n razinu na kojoj se nalazi)

.....

**X1 Header** – zaglavlje grupe najniže razine (X1 označava polje grupiranja najniže grupe)

**Detail** – detaljni podaci unutar najniže grupe, prikazani u ponavljajućim se redovima za svaki podatak mora biti odgovarajuće polje – objekt na izvještaju

**X1 Footer** – podnožje grupe najniže razine (X1 označava polje grupiranja najniže grupe)

.....

**Xn Footer** -

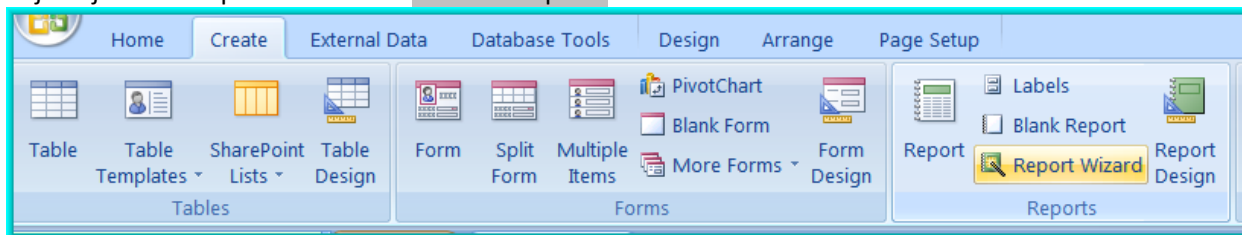
**Page footer** – podnožje stranice; ispisat će se na početku svake stranice izvještaja

**Report Footer** – podnožje izvještaja - pojavljuje se na kraju izvještaja

Svaki od ovih objekata ima karakteristična svojstva kao i ostali objekti (pojedinačne labele i polja).

## 5.4.5 Kreiranje izvještaja

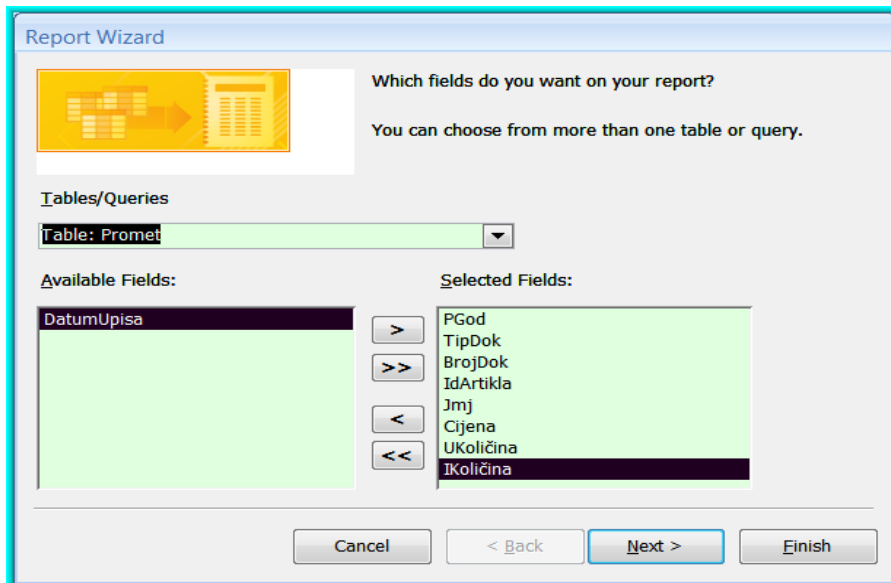
Izvještaj kreiramo preko izbornika **Create->Reports**:

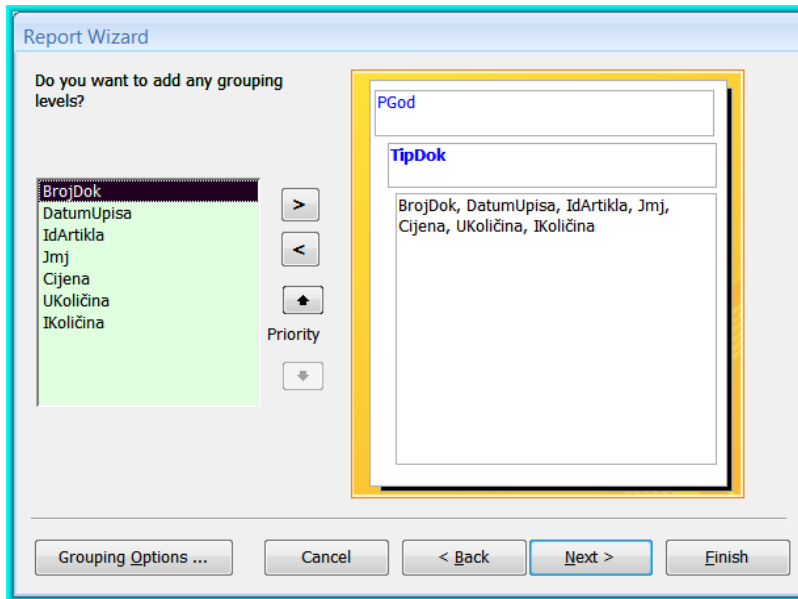


Izvještaj je najjednostavnije započeti kreirati preko ikone „Report Wizard“ (čarobnjak izvještaja). Čarobnjak izvještaja će nas dijaloški voditi kroz postupak kreiranja izvještaja. Nakon toga, možemo direktno podešavati i prilagođavati generirani izvještaj specifičnim potrebama izgleda i funkcionalnosti.

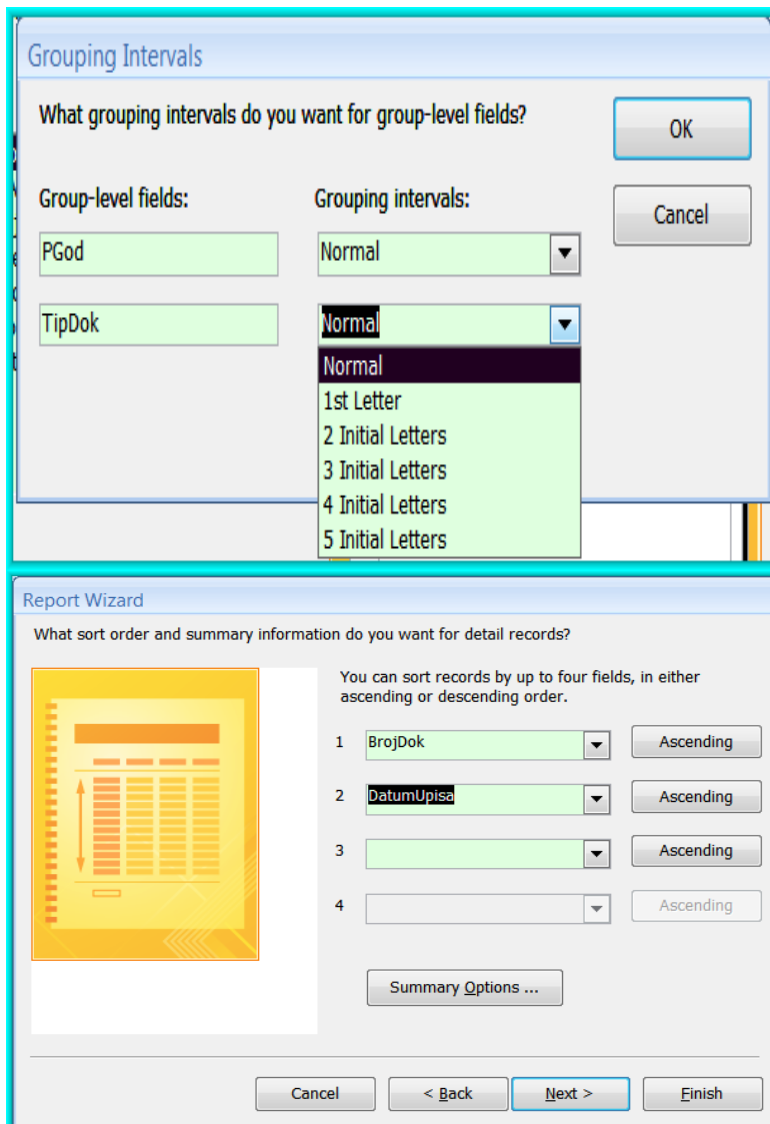
Ako smo započeli kreiranje izvještaja pomoću čarobnjaka izvještaja, najprije treba odabrati tablicu ili pogled kao izvor podataka za izvještaj. Nakon toga, treba odabrati polja izvora (atribute) koji će se na izvještaju vidjeti odnosno koji će se moći uključiti na izvještaj.

Nakon toga će se pojaviti forma na kojoj ćemo definirati hijerarhiju grupiranja podataka. Odabiremo polja prema prioritetu (razini hijerarhije) koji želimo postići.

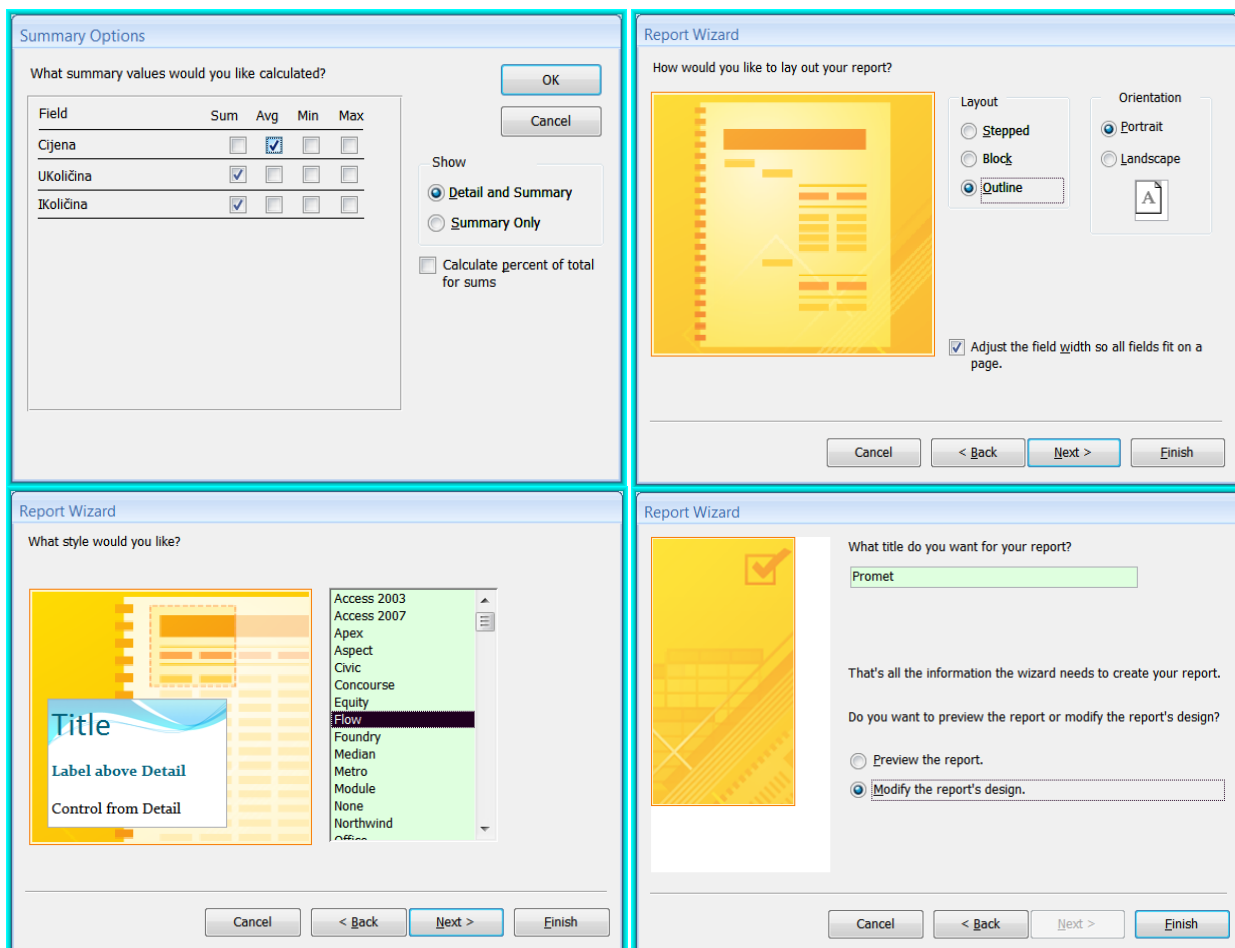




Ako želimo možemo detaljnije definirati način grupiranja. U sljedećoj dijaloškoj formi navodimo redoslijed prikaza podataka iz izvora (sortiranje)



Zatim navodimo za numerička polja iz detalja, sve oblike agregacije (Summary options) za numerička polja. Izabrani agregati će se prikazivati u svakoj grupi. U sljedećem koraku trebamo definirati jedan od ponuđenih izgleda izvještaja s obzirom na prikaz detalji/grupa. Na kraju slijedi izbor stila izvještaja, te završetak generiranja. Možemo završiti dajući izvještaju naziv, s prikazom (Preview) izvještaja ili ići na daljnje uređivanje izvještaja.



Rezultat generiranja će biti izvještaj u design-modu. Sve objekte možemo na njemu mijenjati i prilagođavati, a možemo ih i brisati, premješati i reorganizirati. Također, možemo dodavati nove objekte. Kao i kod formi, i izvještaji mogu sadržavati VBA kod. Značajno je da se programski kod može vezati uz događaje koji će ga pokrenuti (slično kao kod formi). Npr., možemo programirati jednostavnu proceduru koja će prikaz izvještaja maksimizirati prilikom otvaranja izvještaja na ekranu:

```
Private Sub Report_Open(Cancel As Integer)
```

```
    DoCmd.Maximize
```

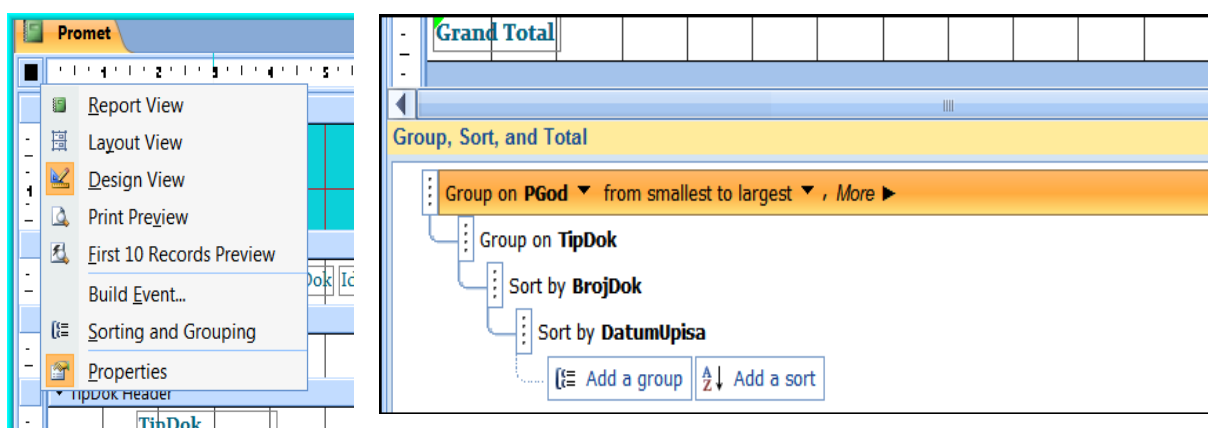
```
End Sub
```



Primjer aktivnog vizualnog oblikovanja izvještaja u Report Designeru:

Report Header									
Pregled prometa									
Page Header									
PGod Header									
PGod									
TipDok Header									
TipDok									
BrojDok	DatumUpisa	IdArtikla	Jmj		Cijena	UKoličina	IKoličina		
Detail									
BrojDok	DatumUpisa	IdArtikla	Jmj		Cijena	UKoličina	IKoličina		
TipDok Footer									
="Summary for " & "TipDok' = " & " & [TipDok] & (" & Count(*) & " & IIf(Count(*)=1,"detail record";"detail records") & ")"									
Sum					=Sum([UKoličina]	=Sum([IKoličina])			
Avg					=Avg([Cijena])				
PGod Footer									
="Summary for " & "PGod' = " & " & [PGod] & (" & Count(*) & " & IIf(Count(*)=1,"detail record";"detail records") & ")"									
Sum					=Sum([UKoličina]	=Sum([IKoličina])			
Avg					=Avg([Cijena])				
Page Footer									
=Now()						="Page" & [Page] & " of " & [Pages]			
Report Footer									
Grand Total					=Sum([UKoličina]	=Sum([IKoličina])			

Ako želimo promijeniti grupiranje, sortiranje ili želimo sakriti/prikazati neku grupu, to postizemo desnim klikom na lijevom vrhu izvještaja. Nakon toga uređujemo ponuđeni prozor.



Izrađeni izvještaj treba spremiti i nakon toga se može aktivirati (Print Preview) ili dalje mijenjati.

Uočimo da se svaki objekt uključujući svaki Header/Footer, može prilagođavati promjenom njegovih svojstava (desni klik + Properties).

## 5.5 Zadaci i sadržaj vježbe

### 5.5.1 Sadržaj vježbe

U ovoj vježbi je potrebno preuzeti ispravnu bazu **Vj-4-2013.accdb**, i preimenovati ju u: **VJ-5-2013.accdb**.

Nakon toga treba izraditi forme sa subformama te izvještaje za svaku tablicu po jedan.

### 5.5.2 Tijek izvođenja vježbe

1. Kreirajte novu bazu Vj-5-2013.accdb prema ispravnoj/dovršenoj bazi iz prošle vježbe
2. Preoblikujte postojeće forme tako da im dodate subforme sa filterom:

Glavna forma	Subforma	Filter
Artikl	Promet	IdArtikl
TipDok	Dokument	TipdDok
PPartner	Promet	IdPP

3. Za svaku formu iz prethodne vježbe izradite po jedan izvještaj. Sami odlučite o izgledu i sadržaju izvještaja. Na forme dodajte dugme „Izvještaj“ pomoću koga će se odgovarajući izvještaj prikazati (Print Preview).

## 5.6 Zaključna pitanja i zadaci

### 5.6.1 Pitanja u vezi izložene vježbe

Po završetku vježbe, studenti bi trebali znati ispravno odgovoriti na sljedeća pitanja (test će se temeljiti na ovim pitanjima):

- Što je subforma, a što glavna forma?
- Kako je moguće povezati subformu s podacima na glavnoj formi?
- Što znači svojstvo LinkMasterFields?
- Što znači svojstvo LinkChildFields?
- Što sve može biti izvor-objekt za subformu?
- Kako se ponaša povezana subforma kad se podaci na glavnoj formi mijenjaju?
- Koliko subformi može biti na jednoj formi?
- Kako je moguće spriječiti/omogućiti mogućnost uređivanja podataka na formi?
- Koje sve vrste izvora podataka možemo imati na formi?
- Što je to izvještaj?
- Koje su glavne razlike između izvještaja i forme?
- Je li moguće formu ispisati na printeru?
- Koji su osnovni koncepti forme?
- Što je Header na izvještaju?
- Što je Footer na izvještaju?
- Kako je moguće grupirati podatke na izvještaju?
- Kojim slijedom će se podaci na izvještaju ispisati?
- Kako je moguće agregirati (sumirati) podatke na izvještaju?
- Kako se izvještaj može ispisati na printeru?
- Kako se određuje format-oblik stranice izvještaja?
- Može li pogled biti izvor podataka za izvještaj?
- Može li tablica biti izvor podataka za izvještaj?
- Kakva je razlika između ova dva oblika izvora podataka za izvještaj?
- Je li moguće izvještaj raditi bez čarobnjaka izvještaja, kako?

### 5.6.2 Zadaci za samostalno produblivanje znanja

Ako vježbu niste do kraja napravili tijekom vježbe, izradite ju naknadno, nakon vježbe. Pokušajte izraditi još koji izvještaj – razmislite na koji bi još način imalo smisla prikazati podatke iz baze.

Ako ne znate odgovore ili ne razumijete pitanja u prethodnom poglavlju, to je znak da vježbu niste savladali – u tom slučaju dodatni napor da ju samostalno savladate. Pitanja na kraju vježbe su pitanja koja su vezana i za ispit – ne preskačite ih olako.

Ako niste u stanju razumijete izloženu materiju ili ako niste u stanju samostalno izraditi vježbu, tražite pomoć od kolega, a ako ni to nije dovoljno, obratite se za pomoć nastavniku.

## 5.7 Literatura i dodatni materijali

- 1) <http://www.gcflearnfree.org/access2010/10>
- 2) <https://support.office.com/en-gb/article/Create-reports-for-a-new-database-cedce547-9f48-4354-8eac-66fb0d4d8656>
- 3) [http://ecdl.hgk.hr/Pdf/W7\\_2010/ITDesk\\_Prirucnik\\_5\\_baze\\_podataka\\_microsoft\\_access\\_2010.pdf](http://ecdl.hgk.hr/Pdf/W7_2010/ITDesk_Prirucnik_5_baze_podataka_microsoft_access_2010.pdf)

## Vježba 6: SQL komande – DDL

### 6.1 Motivacija

Ovom vježbom želimo uvesti studente u SQL jezik, odnosno u dio SQL komandi koje se nazivaju SQL DDL (Data Definition Language). Kroz prethodne vježbe studenti su trebali dobiti uvid u to što je baza podataka, pojam strukture baze podataka i pojam ograničenja. Objekte strukture i ograničenja smo do sada definirali pomoću ACCESS alata. U ovoj vježbi ćemo na postupni način ući u korištenje SQL jezika. Kao i do sada očekujemo dvostruku korist od toga. Prvo, savladati i prakticirati SQL jezik. Drugo, usput naučiti kako se te komande mogu upotrebljavati u Accessu.

#### Sadržaj vježbe:

<b>SQL KOMANDE – DDL</b>	<b>81</b>
6.1	MOTIVACIJA 81
6.2	CILJEVI VJEŽBE – ISHODI UČENJA 82
6.3	SAMOSTALNA PRIPREMA VJEŽBE 82
6.4	TEORIJSKA PRIPREMA VJEŽBE 83
6.4.1	SQL – jezik baza podataka..... 83
6.4.2	Elementi SQL komande..... 84
6.4.3	SQL DDL..... 84
6.4.4	CREATE TABLE komanda ..... 85
6.4.5	ALTER TABLE komanda..... 86
6.4.6	CONSTRAINT klauzula ..... 87
6.4.7	DROP komanda/klauzula ..... 88
6.4.8	Kreiranje SQL DDL komandi u Accessu ..... 89
6.5	ZADATAK I SADRŽAJ VJEŽBE 91
6.5.1	Sadržaj vježbe..... 91
6.5.2	Tijek izvođenja vježbe..... 91
6.6	ZAKLJUČNA PITANJA I DODATNI ZADACI 92
6.6.1	Zadaci za samostalno produbljivanje znanja..... 93
6.7	LITERATURA I DODATNI MATERIJALI 93

## 6.2 Ciljevi vježbe – ishodi učenja

Od studenata se očekuje da, nakon uspješno završene vježbe:

- Shvate/razumiju pojam i namjenu SQL –DDL komandi .
- U stanju su samostalno napisati smislene jednostavnije SQL DDL komande
- Nauče primjenjivati DDL komande u Accessu.
- Razumiju posljedice primjene DDL komandi.

## 6.3 Samostalna priprema vježbe

Prije dolaska u lab, obaveze studenata su:

- Proučite prethodnu materiju sa predavanja iz kolegija BP
- Proučite sve pripremne materijale za ovu vježbu
- Minimalno znanje potrebno za izvođenje vježbe:
  - Teorijski dio ove vježbe: samostalno dobro naučite poglavlje 6.4
  - Koje vrste DML komandi poznajete
  - Što su DML komande
  - Komanda Create
  - Komanda Alter
  - Komanda Drop
  - Što znači CONSTRAINT

## 6.4 Teorijska priprema vježbe

U pripremi za ovu vježbu dobro proučite sljedeći teorijski uvod.

### 6.4.1 SQL – jezik baza podataka

Kao što je iz relacijske teorije podataka poznato, relacijski model podataka mora obuhvatiti tri temeljna aspekta:

- Strukturu - relacije,
- Integritetska pravila ili ograničenja,
- Operacije ili manipulacije podacima.

U prethodnim smo se vježbama bavili kreiranjem strukture i dijela integritetskih ograničenja. To smo radili pomoću u Access ugrađenih alata odnosno tabelarnih predložaka. Važno je da uočimo kako ti alati i pomagala nisu inherentni dio DBMS-a, već se mogu shvatiti kao dodatni alati koji su specifični za Access. Istina, i uz druge DBMS-ove se u pravilu mogu naći odgovarajući dodatni alati sa sličnom namjenom. Međutim, oni ne garantiraju interoperabilnost, tj., ne mogu se koristiti uz ostale DBMS-ove.

Definiranje (kreiranje, promjena) strukture, integritetskih pravila te manipulacije podacima u bazi podataka se mogu realizirati putem specijalnog jezika koji se zove SQL (čitaj: eskveel). SQL nije sastavni dio relacijske teorije. Codd je u svojoj relacijskoj teoriji postavio pravila/zahtjeve koje bi jedan takav jezik trebao ispunjavati, ali ga nije razvio. Najpoznatiji pokušaj da se razvije jedan takav jezik je nastao ranih 70-tih godina dvadesetog stoljeća. To je bio jezik SEQUEL (čitaj: sekvel). SEQUEL se smatra pretečom SQL-a, jer je u IBM Corporation gdje je nastao, ime SEQUEL promijenjeno u SQL. SQL je tijekom 80-tih godina implementiran u IBM-ove DBMS-ove: DB2 i SQL/DS. To je krajem 80-tih godina rezultiralo pojavom ANSI standarda za SQL, koji je praktički preuzeo IBM-ov SQL kao standard. Paralelno s tim drugi su nastojali ući u tržište RDBMS-a uz implementaciju svojih verzija SQL-a. Najznačajniji je Oracle Corporation koji još i danas slovi za jednog od vodećih proizvođača RDBMS-a. Unatoč standardu, različiti su proizvođači iz komercijalnih razloga zadržali izvjesne razlike u odnosu na standard sve do današnjih dana. Zbog toga možemo govoriti ne o jedinstvenom SQL jeziku nego o nizu SQL dijalekata. Potrebno je napomenuti da SQL Standard od početka pa do danas ne propisuje SQL jezik kao strogo relacijski. Vjernost SQL prema standardu u manjoj ili većoj mjeri odstupa od stroge relacijske teorije. Ipak, danas od SQL-a ne postoji bolji odnosno kompletniji jezik za relacijske baze podataka.

## 6.4.2 Elementi SQL komande

SQL je jezik koji u početku osmišljen kao upitni jezik tj., kao jezik koji omogućuje pisanje samostalnih naredbi koje traže neki sadržaj iz baze podataka. Kasnije je SQL postao jezik kojim je moguće ne samo pretraživati podatke u bazi, nego i kreirati i mijenjati podatke i metapodatke (strukturu i ograničenja).

Dakle, SQL jezik nije jezik namijenjen klasičnom programiranju, iako je u proceduralnom SQL-u moguće pisati procedure pa i funkcije. SQL se piše u obliku komandi (statement), pri čemu se svaka komanda izvodi kao cjelina odnosno kao program. Za SQL kažemo da je deklarativan jezik – potrebno je specificirati (deklarirati) što želimo, ali ne i detaljno navesti proceduru postizanja rezultata.

SQL komande se sastoje od više elemenata:

- **Klauzule** – to su ključne riječi preuzete iz engleskog jezika koje imaju točno određeno značenje u komandi.
- **Izrazi** – su kombinacija identifikatora, operatora i konstanti koji daju jednu vrijednost određenog tipa.
- **Operatori** – su oznake koje predstavljaju operaciju nad operandima u nekom izrazu.
- **Identifikatori** – su oznake atributa, tablica, varijabli i ostalih objekata.
- **Konstante** – su konstantne vrijednosti bilo kojeg tipa (tekst, broj,...) .

Ovi elementi s klauzulom na čelu, kombinirani prema pravilima SQL sintakse, čine SQL komande. Svaka SQL komanda završava s točka-zarez znakom (;). Sintaksa SQL-a je propisana standardom, ali u praksi manje ili više odstupa od implementacije do implementacije.

## 6.4.3 SQL DDL

SQL sadrži niz klauzula odnosno komandi. S obzirom na namjenu mogu se razlikovati tri skupine komandi:

- **DDL – data definition language** – komande za definiranje i manipulaciju metapodacima (objektima strukture i ograničenjima)
- **DML – data manipulation language** – komande za manipulaciju s podacima u bazi podataka
- **DCL - data control language** – komande za upravljanje korisnicima, grupama korisnika i njihovim pravima i ovlaštenjima



U ovoj vježbi ćemo obraditi DDL komande i to s primjenom na Ms Access. DDL komande se mogu odnositi na tri akcije:

- CREATE – kreiranje
- ALTER – promjenu
- DROP – Brisanje

#### 6.4.4 CREATE TABLE komanda

CREATE TABLE komanda ima sljedeći pojednostavljeni oblik:

```
CREATE [TEMPORARY] TABLE table (field1 type [(size)] [NOT NULL]
[WITH COMPRESSION | WITH COMP] [index1]
[, field2 type [(size)] [NOT NULL] [index2] [, ...]] [,
CONSTRAINT multifieldindex [, ...]])
```

Riječi pisane malim slovima imaju sljedeće značenje:

Riječ	Opis
<i>table</i>	Naziv tablice koju treba kreirati.
<i>field1, field2</i>	Naziv polja (atributa koje treba kreirati), jedno ili više polja
<i>type</i>	Tip polja
<i>size</i>	Veličina polja u znakovima (za tipove: Text ,Binary )
<i>index1, index2</i>	Constraint- definicija graničenja za jedno indeks od jednog polja
<i>multifieldindex</i>	Constraint- –definicija ograničenja za index sa više polja.

Klazule i njihovo značenje:

Klazula	Opis
CREATE TABLE	Kreiraj tablicu s nazivom tablice
NOT NULL	polje ne može biti null
WITH COMPRESSION ili WITH COMP	Komprimiraj UNICODE polje
CONSTRAINT	ograničenje

Ova komanda omogućuje definiranje/kreiranje nove tablice i svih objekata tablice (atributi/polja, indexi/primarni ključ, ograničenja)

**Primjeri:**

Kreiraj tablicu Kupci s atributima MBKupca (matični broj kupca) tipa Integer, NazivKupca varijabilne dužine 50 znakova, pri čemu uzmi da je MBKupca primarni ključ:

```
CREATE TABLE Kupci (MBKupca INTEGER PRIMARY KEY, NazivKupca NCHAR VARYING (50));
```

Kreiranje tablice Narudzbe s ograničenjem stranog ključa **OSKNarudzbeKupci**, u dvije verzije definicije kaskadnih operacija kod stranog ključa:

```
CREATE TABLE Narudzbe (BrojNar INTEGER PRIMARY KEY, MBKupca INTEGER, Napomena NCHAR VARYING (255), CONSTRAINT OSKNarudzbeKupci FOREIGN KEY (MBKupca) REFERENCES Kupci ON UPDATE CASCADE ON DELETE CASCADE;
```

```
CREATE TABLE Narudzbe (BrojNar INTEGER PRIMARY KEY, MBKupca INTEGER, Napomena NCHAR VARYING (255), CONSTRAINT OSKNarudzbeKupci FOREIGN KEY (MBKupca) REFERENCES Kupci ON UPDATE SET NULL ON DELETE SET NULL;
```

## 6.4.5 ALTER TABLE komanda

Ova komanda omogućuje promjenu definicije već kreirane (postojeće) tablice, odnosno promjenu njene strukture kao što je dodavanje polja, promjene polja, brisanje polja i ograničenja.

```
ALTER TABLE table {ADD {COLUMN field type[(size)] [NOT NULL]
[CONSTRAINT index] |
ALTER COLUMN field type[(size)] |
CONSTRAINT multifieldindex} |
DROP {COLUMN field I CONSTRAINT indexname} }
```

Malim slovima pisane riječi imaju sljedeće značenje:

riječ	opis
<i>table</i>	Naziv tablice čija definicija će se promijeniti.
<i>field</i>	Naziv polja (atributa) koji će se dodati ili promijeniti ili izbrisati iz tablice
<i>type</i>	Tip polja.
<i>size</i>	Veličina polja u znakovima (Text ,Binary ).
<i>index</i>	Indeks za polje.
<i>multifieldindex</i>	Definicija višepoljnog indexa koji će se dodati u definiciju tablice.
<i>indexname</i>	Naziv višepoljnog indexa koji će se izbrisati.

Klauzule i njihovo značenje:

Klauzula	Opis
<i>ALTER TABLE</i>	Promijeni postojeću tablicu s nazivom table
<i>ADD COLUMN</i>	Dodaj novu kolonu (ATRIBUT)
<i>ALTER COLUMN</i>	Promijeni postojeću kolonu
<i>DROP COLUMN</i>	Izbaci/izbriši kolonu

Primjeri:

```
ALTER TABLE Zaposlenici ADD COLUMN Napomena TEXT(25);
```

```
ALTER TABLE Zaposlenici ALTER COLUMN PBR TEXT(10);
```

### 6.4.6 CONSTRAINT klauzula

**CONSTRAINT klauzula** nije samostalna komanda. Ona se može odnositi na ograničenje za jedno ili za više polja. Ova klauzula se može koristiti kao opcionalni dio komande CREATE TABLE ili ALTER TABLE gdje služi za definiranje ograničenja.

Constraint za jedno polje ima sintaksu:

```
CONSTRAINT name {PRIMARY KEY | UNIQUE | NOT NULL |
REFERENCES foreigntable [(foreignfield1, foreignfield2)]
[ON UPDATE CASCADE | SET NULL]
[ON DELETE CASCADE | SET NULL]}
```

CONSTRAINT klauzula za više polja ima sintaksu:

```
CONSTRAINT name
{PRIMARY KEY (primary1[, primary2 [, ...]]) |
UNIQUE (unique1[, unique2 [, ...]]) |
NOT NULL (notnull1[, notnull2 [, ...]]) |
FOREIGN KEY [NO INDEX] (ref1[, ref2 [, ...]]) REFERENCES
foreigntable [(foreignfield1 [, foreignfield2 [, ...]])]
[ON UPDATE CASCADE | SET NULL]
[ON DELETE CASCADE | SET NULL]}
```

Klauzula FOREIGN KEY definira strani ključ. Klauzula REFERENCES označava referentnu tablicu za strani ključ, tj. tablicu u kojoj je strani ključ primarni, pri čemu se navodi naziv referentne tablice i polja koja čine strani ključ.

Klauzula ON određuje ponašanje kaskadnih operacija kod brisanja (DELETE) ili ažuriranja (UPDATE), dok klauzula SET NULL omogućuje nuliranje vrijednosti stranog ključa.

Značenje riječi pisanih malim slovima:

riječ	opis
<i>name</i>	Naziv ograničenja koje treba kreirati
<i>primary1, primary2</i>	Naziv jednog ili više polja koja će sačinjavati primarni ključ.
<i>unique1, unique2</i>	Naziv jednog ili više polja koja će biti ograničena jedinstvenim/neduplikatnim (unique) ključem.
<i>notnull1, notnull2</i>	Naziv jednog ili više polja čije vrijednosti neće moći primati NULL-vrijednost.
<i>ref1, ref2</i>	Naziv jednog ili više polja stranog ključa koja su referencijalno povezana s poljima u drugoj tablici.
<i>foreigntable</i>	Naziv tablice koja sadrži polja navedena u <i>foreignfield</i> .
<i>foreignfield1, foreignfield2</i>	Naziv jednog ili više polja u <i>foreigntable</i> specificirane putem <i>ref1, ref2</i> . Ova se klauzula može ispustiti ako je referencirano polje definirano kao primarni ključ od <i>foreigntable</i> .

Constraint klauzula je slična indexu, razlika je u tome što ona može biti upotrijebljena za uspostavljanje referencijalne veze.

### 6.4.7 DROP komanda/klauzula

Drop komanda služi za izbacivanje/brisanje objekata (metapodataka). Brisati se može tablica, index, procedura i pogled (View). U ovoj vježbi ćemo se ograničiti na brisanje tablice i indexa.

Drop klauzula se može pojaviti nesamostalno kao dio komande ALTER TABLE gdje služi za izbacivanje i brisanje polja (column) ili ograničenja (constraint)

**DROP {TABLE table | INDEX index ON table | PROCEDURE procedure | VIEW view}**

Riječi pisane malim slovima imaju sljedeće značenje:

riječ	opis
<i>table</i>	Naziv tablice koja treba biti izbrisana/odstranjena iz baze ili ime table iz koje treba biti izbrisan indeks
<i>procedure</i>	Naziv procedure koja treba biti izbrisana
<i>view</i>	Naziv pogleda koji treba brisati
<i>index</i>	Naziv indeksa koji treba biti izbrisan iz tablice

*Primjeri:*

*Briši indeks Abc iz tablice Zaposlenici*

*Briši tablicu zaposlenici- briše se definicija i sadržaj*

*Briši pogled NoviZaposlenici*

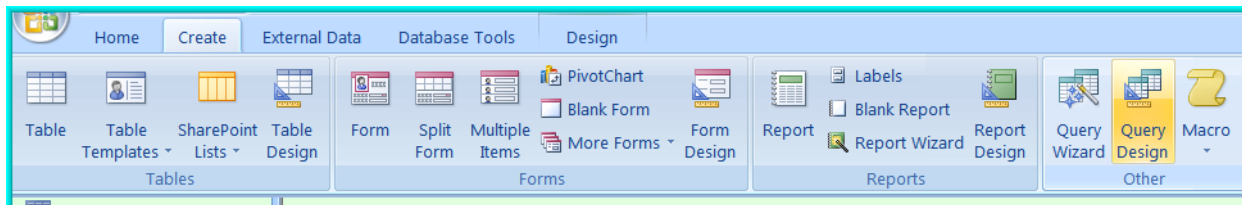
DROP INDEX Abc ON Zaposlenici;

DROP TABLE Zaposlenici;

DROP VIEW NoviZaposlenici

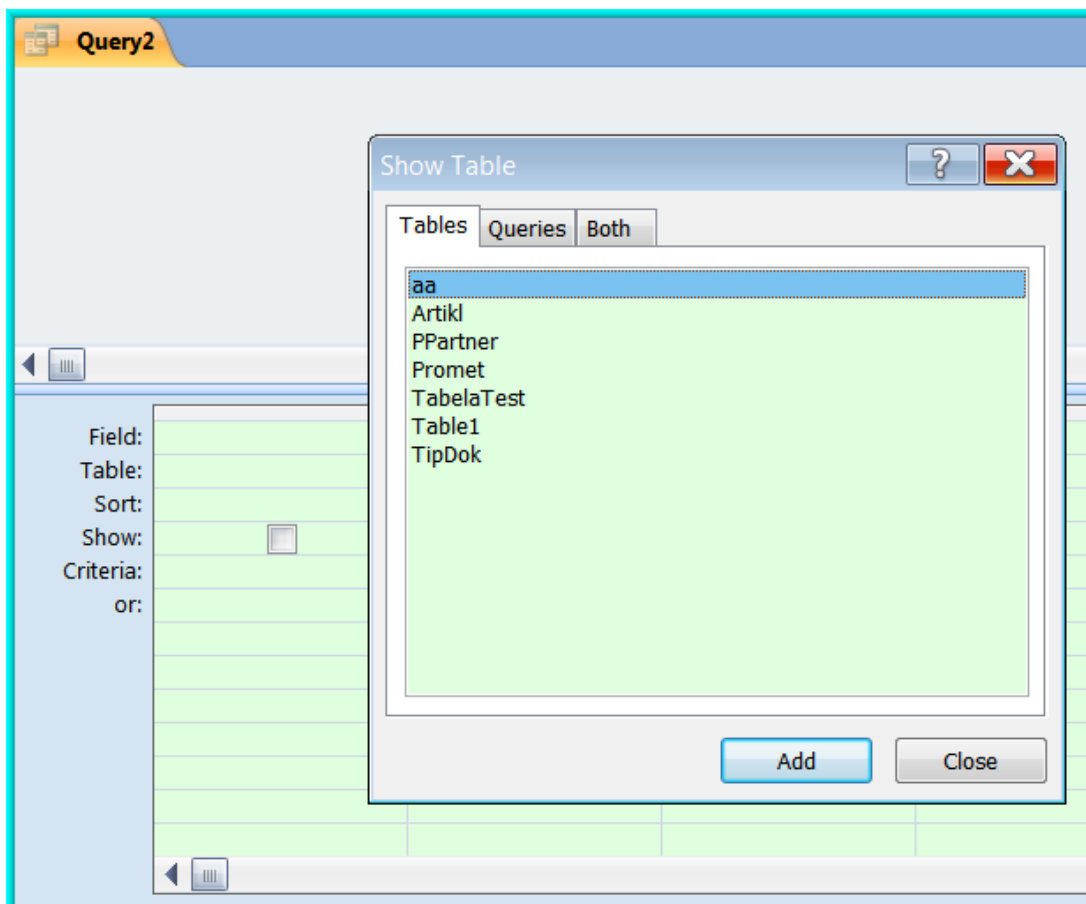
## 6.4.8 Kreiranje SQL DDL komandi u Accessu

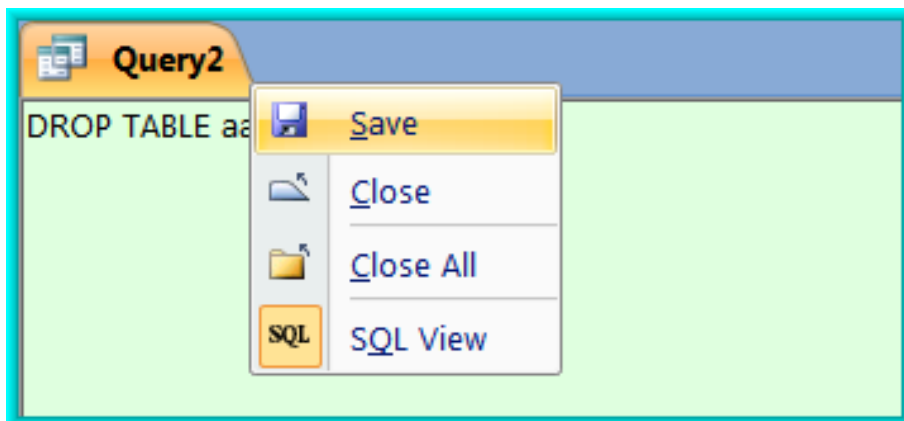
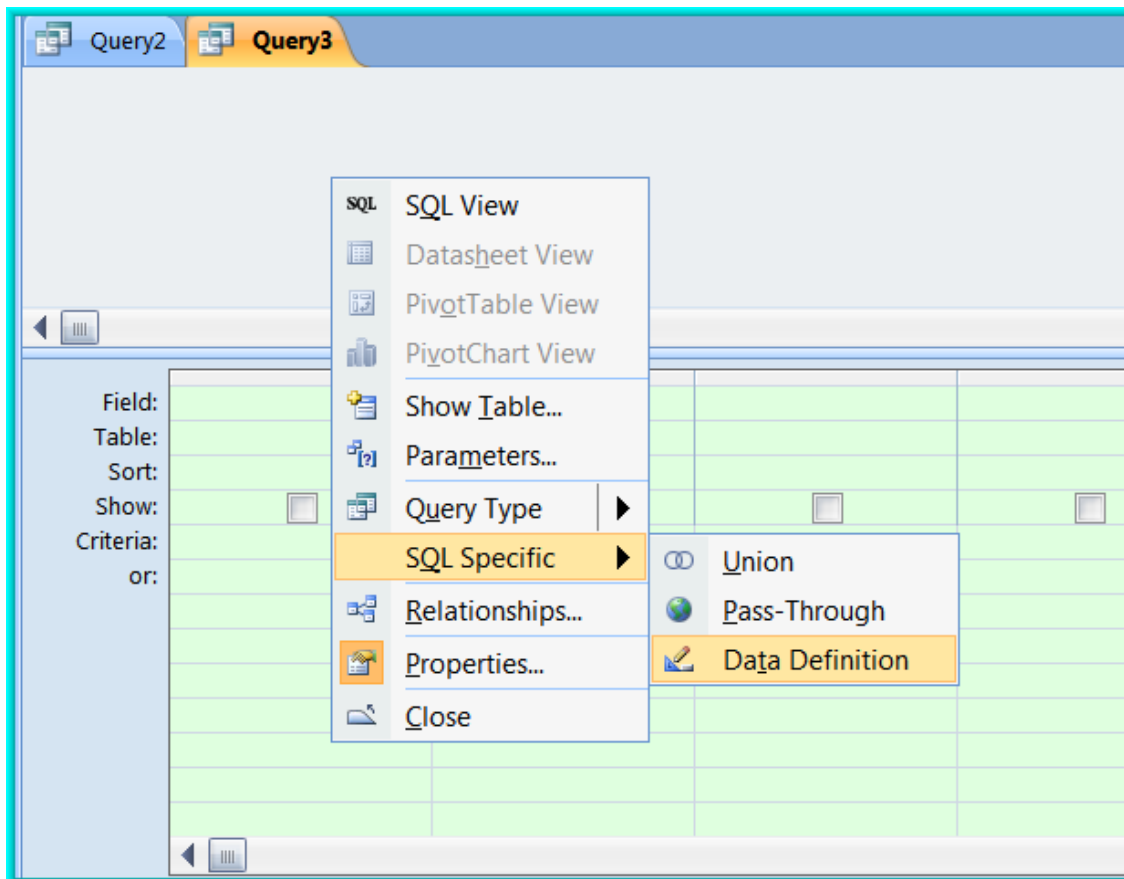
U Accessu se SQL DDL upiti mogu kreirati programskim putem ili putem Query Design alata:



Nakon što se otvori Query prozor potrebno je :

- Zatvoriti prozor Show Table
- Postaviti tip Query-ja na Data Definition (pritisnemo desnu tipku miša u prozoru Query)
- U postavljeni prozor upisati SQL DDL komande
- Spremiti





## 6.5 Zadatak i sadržaj vježbe

### 6.5.1 Sadržaj vježbe

U ovoj vježbi je potrebno otvoriti novu praznu bazu **VJ-6-2013.accdb** te na osnovu baze iz prethodne vježbe definirati sve objekte u novoj bazi pomoću SQL DDL naredbi

Nakon toga treba usporediti shemu novodobivene baze s bazom iz prethodne vježbe. Ako postoje razlike treba ih ispraviti pomoću DDL komandi koristeći Access Query alat.

Ako vježbu ne završite, nastavite ostatak kod kuće.

### 6.5.2 Tijek izvođenja vježbe

- Kreirajte novu bazu Vj-6-2013.accdb.
- Pomoću Query-a upisivanjem SQL DDL komande CREATE TABLE definirajte u njoj svaku tablicu iz prethodne baze Vj-5-2013.accdb.
- Spremite svaki takav Query dajući mu ime DDCimetablice.
- Definirajte za svaku tablicu Query kojim ćete izbrisati dotičnu tablicu- dajte mu naziv DDDimetablice.
- Izbrišite sve definirane tablice pozivajući svaki DDDxxx query.
- Kreirajte sve izbrisane tablice pozivajući svaki DDCxxx query.
- Definirajte potrebna ograničenja uključujući strane ključeve s kaskadnim operacijama.
- Usporedite i analizirajte dobivene tablice sa prethodnom bazom: Vj-5-2013.accdb.
- Ako postoje nesukladnosti ispravite ih također pomoću SQL DDL komandi odnosno definirajući ALTER TABLE query dajući mu ime DDAimetablice.
- Ako ne postoje neusklađenosti iz prethodne točke, definirajte nekoliko DDA xx querya te ih izvedite.
- Ponovo izvedite DDD, a nakon toga i DDC query za tablice koje ste promijenili u točki 10.

## 6.6 Zaključna pitanja i dodatni zadaci

Po završetku vježbe, studenti bi trebali bi znati ispravno odgovoriti na sljedeća pitanja (test će se temeljiti na ovim pitanjima):

- Što je SQL?
- Što je SQL standard?
- Što je SQL dijalekt?
- Od kojih elemenata se sastoji SQL komanda?
- Što je klauzula?
- Što je izraz u SQL-u?
- Što je operator?
- Koje grupe komandi postoje u SQL-u?
- Što znači DDL?
- Koje su osnovne komande DDL-a?
- Što znači klauzula CREATE TABLE?
- Što znači klauzula DROP?
- Što znači klauzula DROP TABLE?
- Što znači klauzula ALTER TABLE?
- Što znači klauzula CONSTRAINT?
- Kako se definira referencijalno ograničenje ?
- Kako se definira strani ključ?
- Kako je moguće promijeniti primarni ključ?
- Kako je moguće izbrisati polje iz tablice?
- Kako je moguće promijeniti tip nekog polja u tablici?
- Kako je moguće ograničiti da neko polje u tablici ne može primiti NULL vrijednost?
- Kako je moguće definirati ograničenje jedinstvene vrijednosti (unique) za neko polje?
- Kako je to moguće učiniti za više polja?
- Što je indeks?
- Kako se u Access Query-u definira DDL query?



### 6.6.1 Zadaci za samostalno produblјivanje znanja

Nakon što ste završili vježbu u labu, pokušajte pomoću SQL DDL komandi mijenjati pa opet popravljati objekte u bazi. Provjerite pomoću neposrednog uvida u tablice da li su promjene onakve kakve ste očekivali da bi trebale biti. Ako nisu, konzultirajte se s kolegama ili pitajte nastavnika ako sami ne možete riješiti problem.

## 6.7 Literatura i dodatni materijali

- 1) <https://support.office.com/en-au/article/Create-or-modify-tables-or-indexes-by-using-a-data-definition-query-d935e129-229b-48d8-9f2d-1d4ee87f418e?ui=en-US&rs=en-AU&ad=AU>
- 2) <http://sql.org/sql-database/sql-tutorial/>
- 3) <http://www.w3schools.com/sql/>



## Vježba 7:

## ODBC

## 7.1 Motivacija

Kao što je već objašnjeno u početnim vježbama, standardno korisničko sučelje prema bazi podataka su aplikacijski programi. Da bi neki aplikacijski program mogao komunicirati s određenom bazom podataka, potrebna je odgovarajuća programska podrška specifična za dotični DBMS. Takva podrška obično u formi knjižnice rutina (library), sadrži odnosno skriva specifičnost konkretnog DBMS-a, a obično se naziva API. U aplikacijski program je čvrsto ugrađeno sučelje prema bazi podataka na odgovarajućem API-ju. Za pristup bazi podataka s istom shemom, ali u nekom drugom DBMS-u, potrebno je napraviti drugi program s karakterističnim API-jem za dani DBMS.

Prema tome fleksibilnost korištenja različitih DBMS-a za isti program u pravilu nije moguća. Ipak, uz izvjesna ograničenja, postoji univerzalno rješenje za pristup različitim DBMS sustavima, koje se zove ODBC. U ovoj vježbi će se predstaviti pojam ODBC-a kao standarda za pristup relacijskim bazama podataka u Windows okruženju preko ODBC API standardnog programskog sučelja.

## Sadržaj vježbe:

<b>VJEŽBA 7:</b>	<b>ODBC</b>	<b>95</b>
7.1	MOTIVACIJA	95
7.2	CILJEVI VJEŽBE – ISHODI UČENJA	96
7.3	SAMOSTALNA PRIPREMA VJEŽBE	96
7.4	TEORIJSKI DIO PRIPREME VJEŽBE	97
7.4.1	ODBC – Open Database Connectivity .....	97
7.4.2	ODBC arhitektura .....	98
7.4.3	Arhitektura Ms Accessa .....	98
7.4.4	ODBC veza s SQL Server SUBP .....	100
7.5	SADRŽAJ I ZADATAK VJEŽBE	105
7.5.1	Tijek izvođenja vježbe .....	105
7.6	ZAVRŠNA PITANJA I ZADACI	106
7.6.1	Završna pitanja u vezi izložene materije vježbe .....	106
7.6.2	Zadaci za dodatno samostalno produblivanje znanja .....	106
7.7	LITERATURA I DODATNI IZVORI	106

## 7.2 Ciljevi vježbe – ishodi učenja

Od studenata se očekuje da nakon uspješno završene vježbe:

- Shvate pojam i namjenu ODBC-a.
- Nauče definirati ODBC konekcije prema različitim izvorima podataka.

## 7.3 Samostalna priprema vježbe

Obaveza studenata je pripremiti se za aktivno sudjelovanje u izvođenju vježbe:

- Proučite prethodnu materiju izloženu na predavanjima iz kolegija BP.
- Dobro proučite teorijski dio ove vježbe.
- Potražite dodatne informacije na internetu kojima možete produbiti potrebno znanje.
- Osim teorijske podloge za pripremu ove vježbe, pokušajte shvatiti i ostali dio materijala.
- Minimalno potrebna znanja za izvođenje vježbe:
  - Pojam ODBC, što rješava
  - Arhitektura ODBC-a
  - ODBC i Access
  - ODBC i povezivanje SQL Servera

## 7.4 Teorijski dio pripreme vježbe

### 7.4.1 ODBC – Open Database Connectivity

ODBC je aplikacijsko programsko sučelje (API) za pristup relacijskim i nerelacijskim DBMS sustavima putem SQL standarda. Osnovna prednost ovog pristupa leži u interoperabilnosti koja programerima omogućuje izradu programski nespecifičnih aplikacija zasnovanih na relacijskim bazama podataka kojima se potom pristupa odabirom DBMS specifičnih upravljačkih programa. U tradicionalnom smislu, aplikacije su namijenjene izvršavanju specifičnih zadataka nad bazom podataka pod nadležnosti dedikiranih sustava za upravljanje bazama podataka (SUBP). Takve su aplikacije građene principom ugrađenog SQL koda koji zahtijeva rekompilaciju svakim novim prijelazom u druga softverska i hardverska radna okruženja. ODBC s druge strane pruža odvojeni programski dio kojemu je zadaća omogućiti aplikacijama pristup podacima unaprijed nepoznatih baza podataka putem standardiziranog sučelja posredstvom dostupnih biblioteka upravljačkih programa. ODBC kao standardno sučelje obuhvaća sljedeće komponente:

- biblioteku dostupnih funkcija kojima se aplikacije povezuju na specifične DBMS-ove, izvršavaju izraze te dohvaćaju rezultate izvođenja tih izraza
- SQL sintaksu definiranu X/Open i SQL Access Group (SAG) CAE (Common Applications Environment) specifikacijom. Specifikacije X/Open organizacije namijenjene su industrijskoj primjeni u hardverski neovisnim okruženjima s ciljem povećanja portabilnosti i interoperabilnosti aplikacija.
- Standardni skup kodnih grešaka
- Standardni način povezivanja i autorizacije s DBMS-om
- Standardnu definiciju podatkovnih tipova

Isti ODBC funkcijski pozivi koriste se u pristupu različitim aplikacijama s razlikom u upravljačkim programima kojima se pozivi pretvaraju u kontekst baza podataka na koje se pozivaju. Primjerice, da bi neka aplikacija imala osiguran pristup Oracle bazama podataka, potrebni su ODBC upravljački programi kao medijator pri radu s odabranom bazom. No, u samoj aplikaciji nije potrebno navoditi na koji će se sve SUBP aplikacija povezivati, već je potrebno definirati ODBC pozive procedura za spajanje s bazama podataka te izvođenje SQL komandi i manipulaciju podacima. Time je omogućena aplikativna interoperabilnost i portabilnost na korisničkoj strani jer je jednostavnim naknadnim povezivanjem aplikacije s odgovarajućim ODBC upravljačkim programom za željeni SUBP osigurana i omogućena komunikacija aplikacije s odabranim SUBP-om.

## 7.4.2 ODBC arhitektura

Prva inačica (verzija) Accessa pojavila se 1992. godine. Temelj na kome je on razvijen bio je vrlo ozbiljan. Microsoft je 80-tih godina pokrenuo projekt „Omega“ čiji je cilj bio razvoj tržišno konkurentnog relacijskog sustava za upravljanje bazama podataka (RDBMS). Budući da je Microsoft kasnije od tog sustava odustao, dobivena tehnologija bila je upotrijebljena za razvoj Accessa. U to vrijeme Microsoft je favorizirao BASIC programski jezik „kao programski jezik za mase“ pa je logična posljedica bila da se on uključi u Access projekt. Pojava Accessa nije imala dovoljno pogodno tlo, budući da Microsoft nije imao dovoljno dobar 32-bitni operativni sustav. Pojava Windowsa95 i nakon toga Microsoft Office-95 paketa čiji je Access bio dio, značilo je prekretnicu u tržišnom nastupu Accessa.

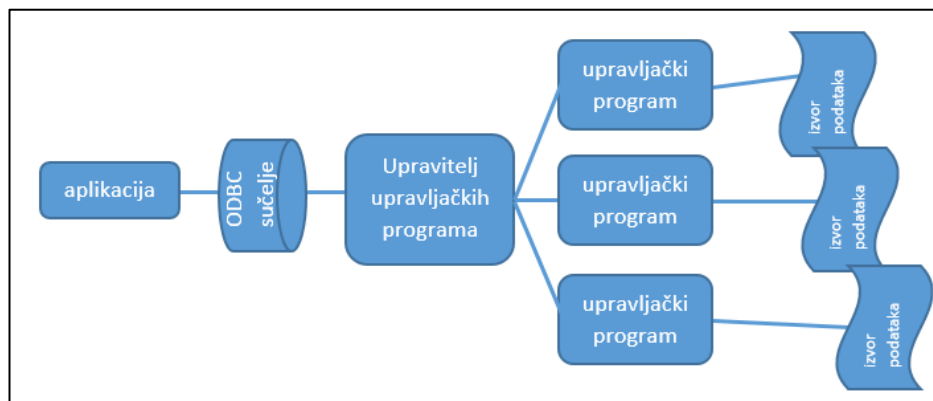
Sastavni dio paketa i glavni programski jezik Accessa je bio Visual Basic for Applications (VBA). VBA je je prošireni podskup Visual Basica, koji je Accessu omogućio stvaranje procedura koje inače nisu bile moguće u jednostavnijoj verziji SQL-a koje Access podržava. Nakon Verzije 95, pojavila se inačica 97. Međutim, sljedeću prekretnicu je predstavljala inačica Access 2000. Uslijedile su inačice Access XP i Access 2003. Ove inačice (verzije) su bile zasnovane na Jet DBMS-u (Jet Engine).

Od inačice 2007, preko 2010 pa do zadnje inačice 2013, Jet DBMS je napušten, a njegova arhitektura je promijenjena. Promijenjena je i razvojna okolina u skladu s Office konceptom. Posljednja inačica je usredotočena na web aplikacije korištenjem Microsoftovog Sharepoint servisa a u osnovi bi SQLServer trebao biti DBMS . Njena desktop verzija je u velikoj mjeri kompatibilna s inačicom 2007 odnosno 2010. Za ove vježbe smo odabrali 32-bitnu inačicu Access 2010, prije svega zbog kompatibilnosti sa prethodnom verzijom OS Windows xp, kako bi bila dostupnija studentima. Inačice Accessa počevši od 2007, mogu podržavati 32 ili 64-bitno adresiranje. 64-bitna verzija zahtijeva 64-bitni Windows operativni sustav, dok 32-bitna verzija može raditi i pod 32-bitnim, ali pod 64-bitnim Windowsima.

## 7.4.3 Arhitektura Ms Accessa

Arhitektura ODBC-a sastoji se od četiri osnovne komponente (slika 1):

- Aplikacije,
- Upravitelja upravljačkih programa,
- Upravljačkih programa te
- Izvora podataka.



Slika 7.1. ODBC arhitektura

Preko standardnog ODBC sučelja aplikacije pozivaju ODBC funkcije s ciljem izvršenja SQL komandi nad određenom bazom podataka. Kroz ODBC sučelje aplikacije zatražuju konekciju ili sjednicu s izvorom podataka, šalju SQL upite izvoru podataka, definiraju i rezerviraju podatkovni prostor i podatkovne tipove za rezultate SQL upita, zatražuju rezultate SQL upita, obrađuje pogreške, izvještavaju korisnike o rezultatima upita, završavaju svaku transakciju s COMMIT ili ROLLBACK komandama za kontrolu transakcija, zatvaraju konekciju ili sjednicu s izvorom podataka.

Upravitelj upravljačkih programa po potrebi učitava upravljačke programe u ime aplikacija koje ih zahtijevaju da bi mogle komunicirati s odabranim SUBP-om. Microsoft isporučuje upravitelje ODBC upravljačkih programa u obliku dinamički povezanih biblioteka (DLL). Osim dinamičkog učitavanja ODBC upravljačkih programa, provodi se mapiranje naziva odredišnih baza podataka s odgovarajućim bibliotekama funkcijskih poziva upravljačkih programa, istovremeno prikupljaju i ocjenjuju parametre višestrukih ODBC inicijalizacijskih poziva prema različitim izvorima podataka.

Upravljački programi obrađuju funkcijske pozive ODBC-a i prevode ih u SQL komande specifične za pojedini izvor podataka (SUBP) te vraćaju aplikacijama rezultate izvođenja komandi. To su dijeljene biblioteke u kojima se implementiraju ODBC funkcijski pozivi modifikacijom u poznate formate odredišnih baza podataka. Aplikacijama se osim rezultata izvođenja upita nad bazom podataka dostavljaju i eventualno obrađene pogreške koje su prethodno preuzete i prevedene u standardne kodove pogrešaka.

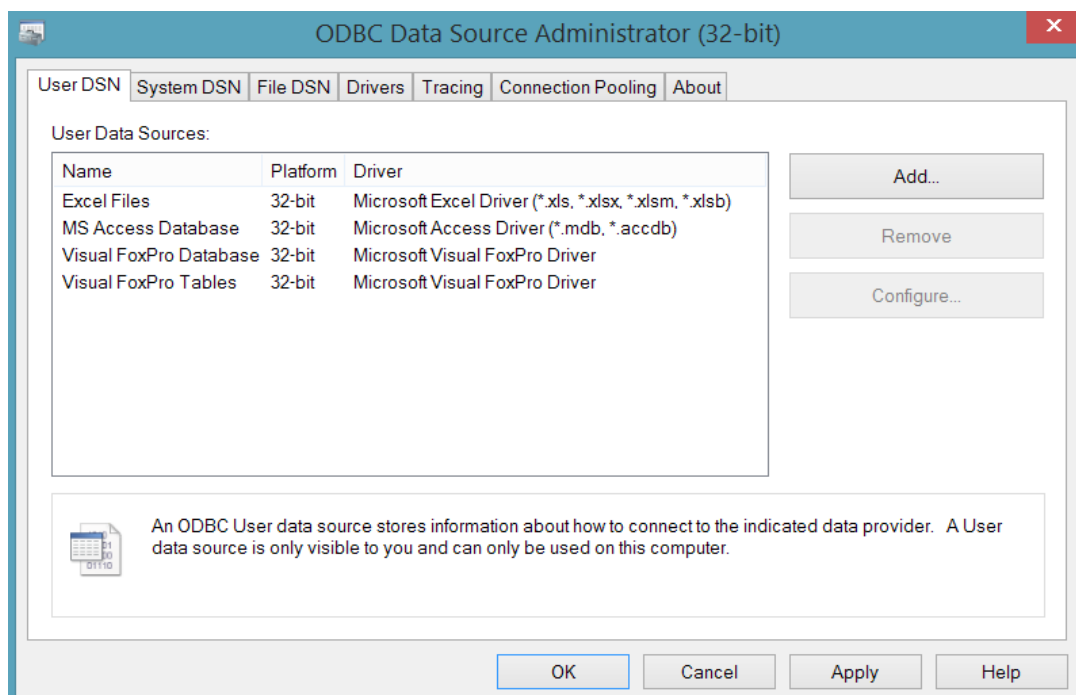
Izvor podataka predstavlja kombinaciju odredišnog DBMS-a/tekstualne datoteke/CSV datoteke/Excel datoteke kojemu ili kojoj se želi pristupiti putem aplikacije, odredišnog operacijskog sustava koji udomljuje odredišni DBMS/..., te resursa potrebnih za pristup udaljenom odredištu koje može biti smješteno na istom računalu kao i aplikacija ili na nekoj udaljenoj lokaciji (mrežna lokacija, naziv baze podataka, korisničko ime i lozinka i sl.)

Da bi zahtjevi aplikacija bile usklađene s mogućnostima upravljačkih programa mora postojati razrađena hijerarhija razina usklađenosti. Razina usklađenosti između aplikacija i upravljačkih programa

određuje skup funkcionalnosti udaljenih DBMS-a dostupnih aplikacijama. ODBC sučelje definira usklađenost upravljačkih programa u dvije skupine, svaka u tri razine usklađenosti: ODBC API s osnovnom, prvom i drugom razinom usklađenosti te SQL gramatiku u koju su uključene i definicije podatkovnih tipova definiranu kroz minimalnu, osnovnu i proširenu razinu usklađenosti. Da bi neki upravljački program zadovoljavao određenu razinu usklađenosti sa zahtjevima funkcionalnosti od strane aplikacija potrebno je zadovoljiti sve funkcionalnosti dotične razine funkcionalnosti, neovisno jesu li tražene funkcionalnosti podržane konkretnim DBMS-om. Svi ODBC upravljački programi moraju zadovoljavati minimalni skup SQL gramatike koju čini podskup osnovne sintakse SQL-92 standarda (CREATE TABLE, DROP TABLE, SELECT, INSERT, UPDATE, jednostavni izrazi te podatkovni tipovi CHAR, VARCHAR, LONG VARCHAR).

#### 7.4.4 ODBC veza s SQL Server SUBP

Da bi se ostvarila prikazana veza potrebno je na računalu s izvorom podataka instalirati upravljački program (MS SQL ODBC Driver) te definirati podatkovni okvir poznat kao DSN – Data Source Name kojim se opisuju detalji konekcije s izvorom podataka. Navedeni koraci obavljaju se putem ODBC upravitelja upravljačkih programa (ODBC Data Source Administrator) do kojega se u Windows okruženju dolazi putem liste Administrativnih alata (Control Panel -> Administrative Tools -> ODBC Data Source Administrator).

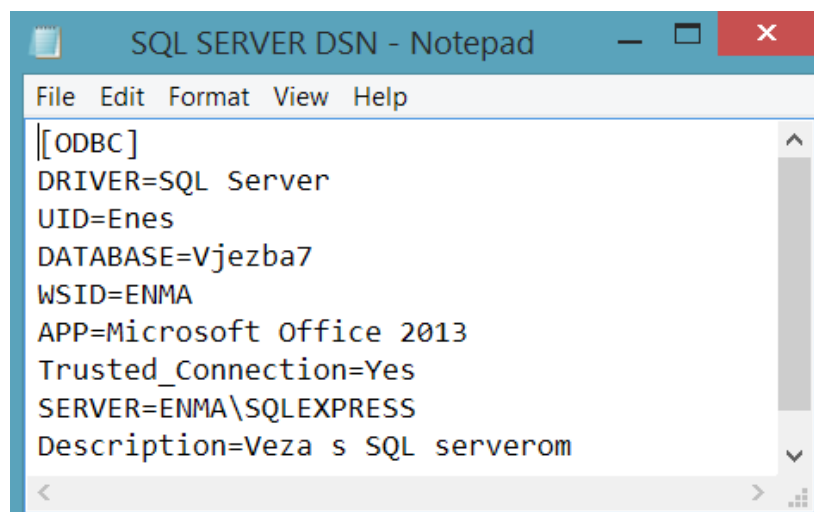


Slika 7.2. Windows alat za povezivanje ODBC upravljačkih programa s izvorom podataka (Upravitelj upravljačkih programa)

Naziv izvora podataka može biti definiran na sistemskoj ili korisničkoj razini kao i putem DSN tekstualne datoteke. Primjer sadržaja DSN datoteke prikazan je slikom 4. Sadržaj DSN datoteke varira



ovisno o tipu izvora podataka, a najčešći atributi koji ga opisuju su naziv i lokacija izvora podataka, naziv upravljačkih programa koji će pristupati izvoru podataka, korisničko ime i lozinka za pristup izvoru podataka.

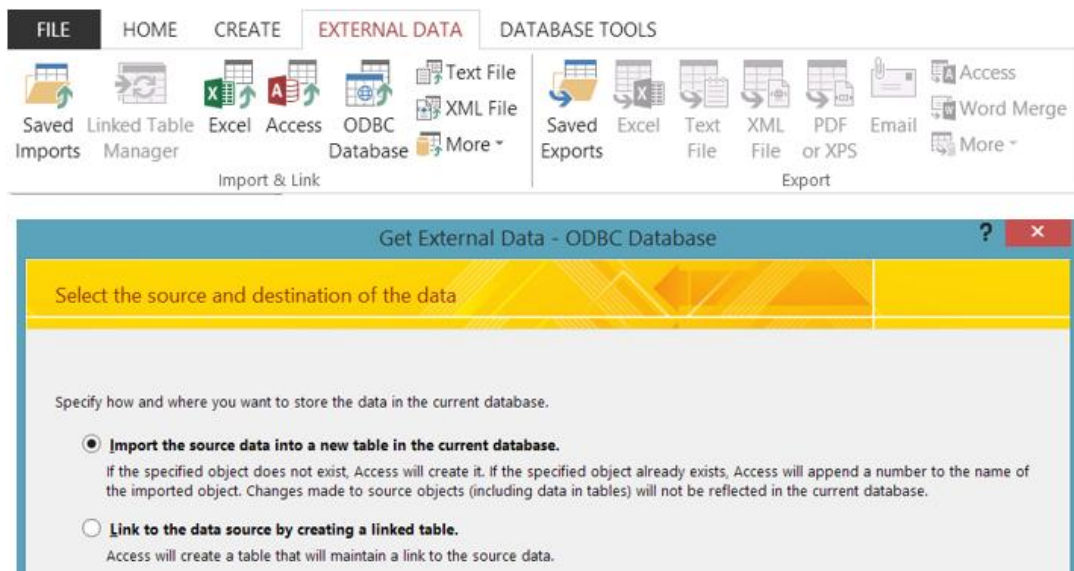


Slika 7.3. Sadržaj DSN datoteke

Korisnički DSN dostupan je samo lokalno prijavljenim Windows korisnicima, za razliku od sistemskog DSN-a koji je dostupan svim korisnicima Windows operacijskog sustava na računalu. Oba DSN-a koriste podatke spremljene u zajedničkim konfiguracijskim datotekama (/etc/odbc.ini, ~/.odbc.ini) ili u registru lokalnog računala (Windows Registry HKLM\Software\ODBC\odbc.ini), a konfiguriraju se putem čarobnjaka unutar alata ODBC Data Source Administrator.

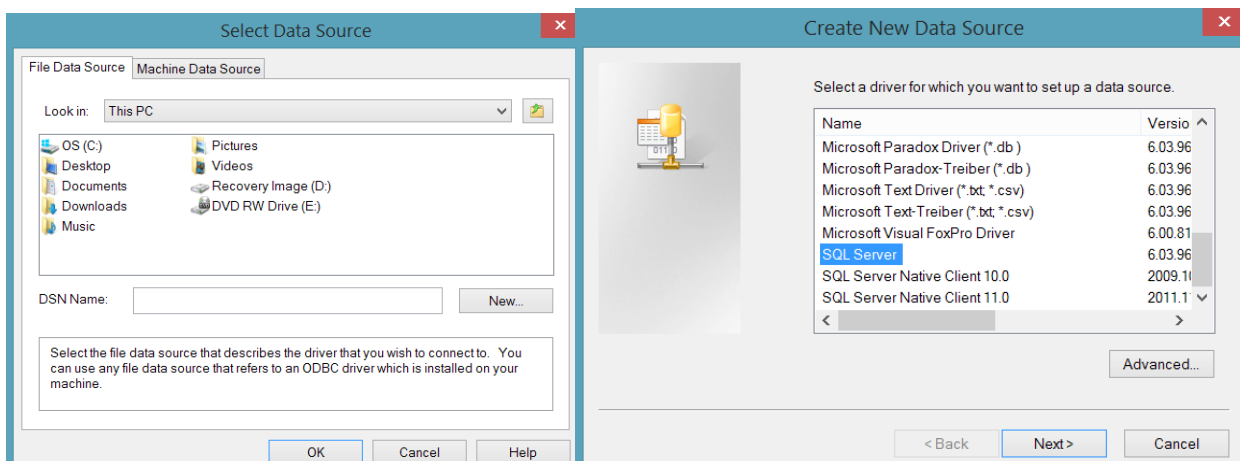
Access se s SQL Serverom može povezati s namjerom trajnog prebacivanja tablica iz SQL Server baze podataka ili kreiranje pogleda na tablice u SQL Server bazi podataka s kojima je moguće vršiti manipulaciju kao da su lokalno spremljene na lokaciji aplikacije. U svrhu ove vježbe pokazat će se povezivanje Access 2010 baze podataka s bazom podataka u SQL Server R2 Express okruženju.

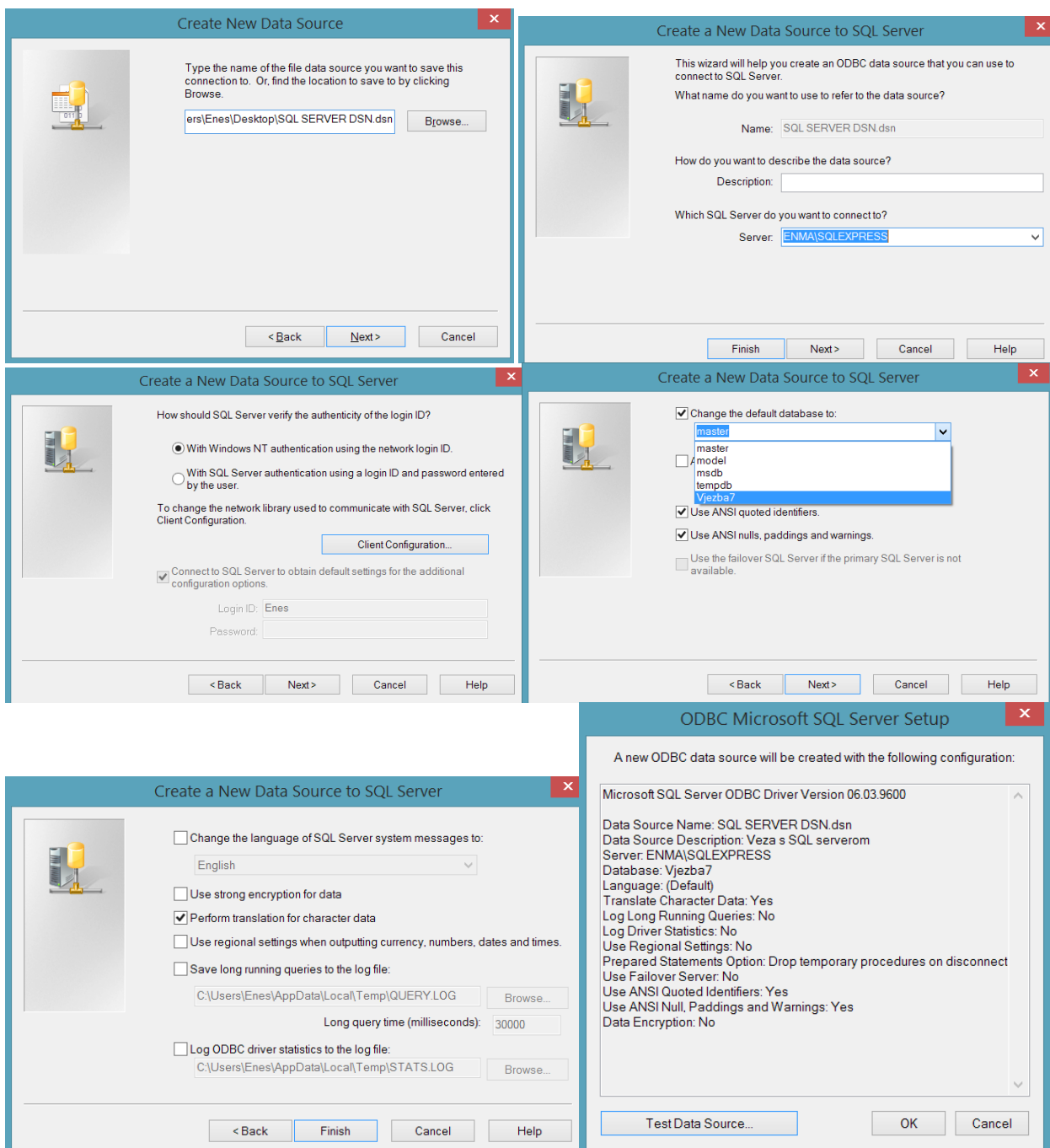
Prije početka povezivanja potrebno je pokrenuti Access s administratorskim ovlastima (Run as administrator) te kreirati novu bazu podataka. ODBC konekcija prema SQL Server DBMS-u ostvaruje se pokretanjem čarobnjaka ODBC Database smještenog u okviru kartice External Data u alatnoj traci MS Accessa. Pokretanjem čarobnjaka otvara se početni dijaloški okvir u kojemu se može definirati žele li se podaci koje sadržava izvor na kojega se aplikacija spaja spremiti lokalno u okruženje aplikacije ili im se želi pristupiti putem posebno definiranih povezanih (linkanih) tablica koje predstavljaju logičku vezu sa stvarnim tablicama smještenim na nekoj udaljenoj lokaciji (npr. unutar SQL Server baze podataka). U slučaju izbora druge opcije (*Link to the data source by creating a linked table*), podaci na udaljenoj lokaciji će se moći pregledavati i mijenjati iz Access okruženja. Nakon potvrde ove opcije otvara se prozor unutar kojega je potrebno odabrati opciju Datotečni izvor podataka (*File Data Source*) te klikom na dugme New pokrenuti dijaloški okvir za definiranje ODBC upravljačkog programa za rad s SQL Server bazom podataka.



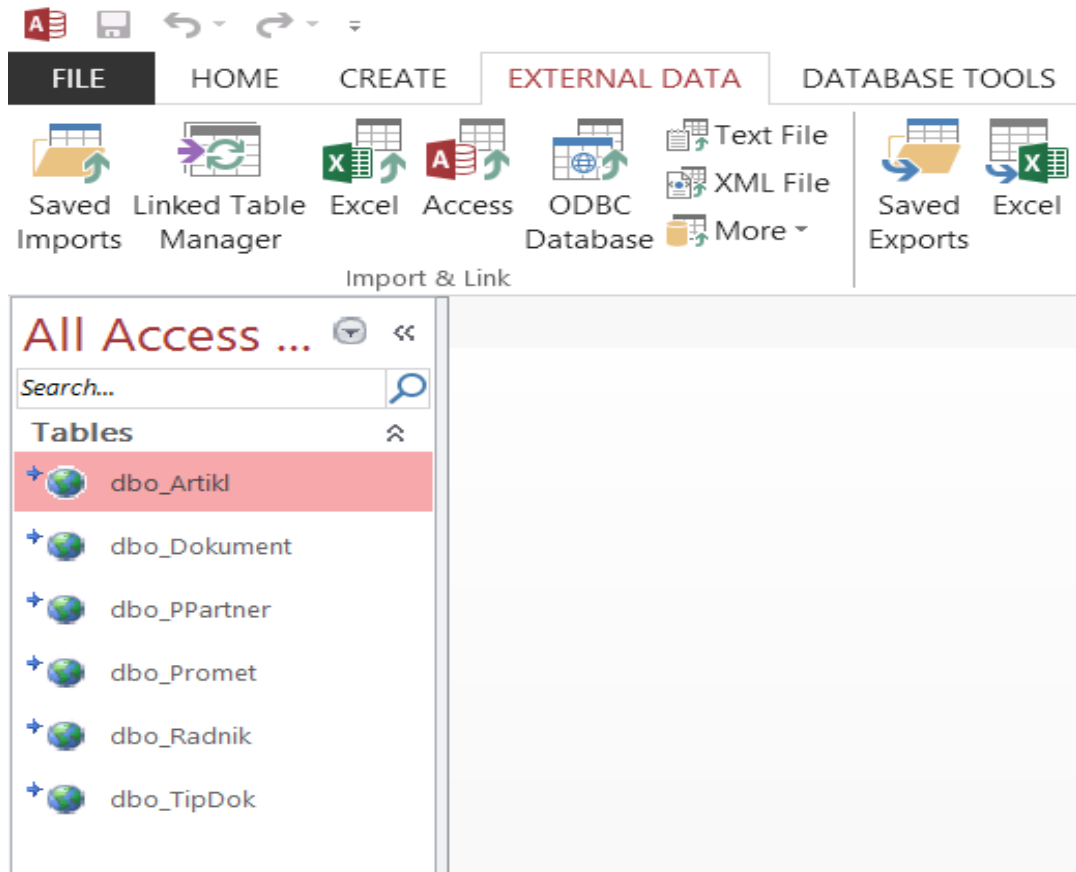
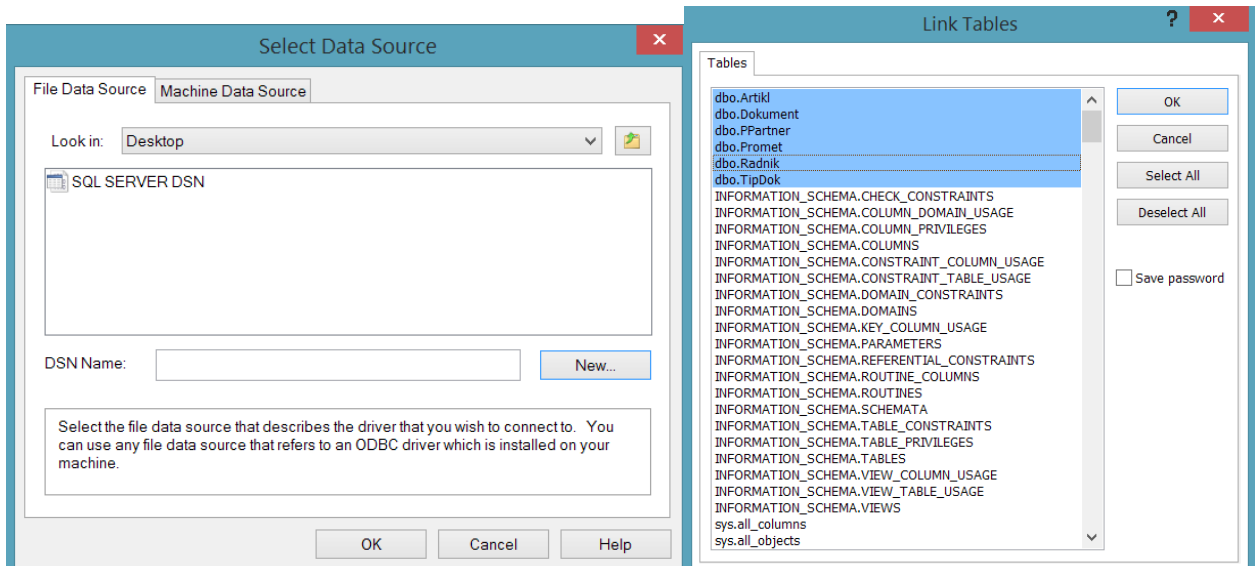
Slika 4 Pokretanje Access čarobnjaka za kreiranje ODBC veze s vanjskim izvorom podataka

Potvrdom SQL Server upravljačkog programa pritiskom na dugme Next pokreće se procedura za kreiranje konekcije izvora podataka s upravo odabranim ODBC upravljačkim programom. U prvom prozoru potrebno je definirati putanju do DSN datoteke koja će se završetkom procedure napuniti podacima potrebnim za ODBC konekciju. Primjer pokazuje da je na radnoj površini korisnika kreirana prazna DSN datoteka (ekstenzija .dsn) nazvana SQL SERVER DSN. U sljedećem koraku, nakon klika na dugme Next, otvara se dijaloški okvir Create a New Data Source to SQL Server u kojemu je potrebno definirati izvor podataka (**Server: <naziv računala> \<naziv instance SQL Servera>**). U sljedećem se koraku definiraju korisnički podaci potrebni za pristup serveru (SQL Server korisničko ime i lozinka ukoliko postoji ili Windows metoda autentikacije) te mrežne postavke (mrežni protokol i prateći parametri, alias servera te njegovo ime). U sljedećem koraku odabire se baza podataka za koju se konekcija definira, a odabire se putem padajućeg izbornika s listom dostupnih SQL Server baza podataka. Na posljednjoj stranici čarobnjaka potrebno je potvrditi postupak klikom na dugme Finish. Nakon toga se pojavljuje sažetak definirane konekcije te mogućnost njenog testiranja.





Završetkom kreiranja ODBC konekcije s izvorom podataka, pojavljuje se upravo definirana konfiguracijska datoteka prethodno nazvana SQL SERVER DSN koju je potrebno označiti te potvrditi klikom na dugme OK. Nakon ove akcije otvara se prozor Link Table u kojemu se nalazi popis svih tablica iz prethodno odabranog izvora podataka. Potrebno je istovremeno označiti sve tablice iz baze podataka Vjezba7 (dbo\_Artikl, dbo\_TipDok, dbo\_Dokument, dbo\_PPartner, dbo\_Radnik, dbo\_Promet). Nakon potvrde označenih tablica, klikom na dugme OK, pojavljuju se ikonice povezanih tablica u navigacijskom oknu Accessa, a sadržaj popisanih tablica moguće je provjeriti dvoklikom na željenu tablicu.



## 7.5 Sadržaj i zadatak vježbe

### 7.5.1 Tijek izvođenja vježbe

U ovoj vježbi je potrebno otvoriti novu praznu bazu **VJ7bp.accdb**, te se povezati na dva različita izvora podataka posredstvom ODBC konekcije.

1. Pokrenite Access s administratorskim ovlastima te kreirajte praznu bazu podataka Vj7bp.accdb
2. Sljedeći upute opisane ovom pripremom te uputama u Prilogu povežite se s SQL Server 2008 R2 bazom podataka Vjezba7.
3. U Accessu kreirajte obrasce za svaku tablicu kao što ste naučili u vježbi 4.
4. Unesite po jednu novu n-torku u tablicu PPartner i Artikl putem upravo kreiranih obrazaca.
5. Provjeriti sadržaj tablica PPartner i Artikl u SQL server 2008 R2 bazi podataka Vjezba7. Uočite postoje li kakve promjene. Koristite funkciju Refresh te ponovo provjerite ima li promjena.
6. Unesite još jednu n-torku u tablici PPartner u SQL Server 2008R2 bazi podataka Vjezba7. Provjerite sadržaj tablice PPartner u Accessovoj bazi podataka Vj7bp. Uočite postoje li kakve promjene. Koristite funkciju Refresh te ponovo provjerite ima li promjena.
7. Kreirajte jednostavan pogled koristeći alat QBE (Query By Example). Neka se prikažu sve prometne stavke dokumenata za koje vrijedi da su cijene artikala veće od 10kn te ulazna, odnosno izlazna količina veća od 50.
8. Pokušajte kreirati ODBC konekciju Accessa s Excel datotekom priloženu s vježbom (preuzmite datoteku TestniPodaci.xls te ju spremite na odabranu lokaciju na vašem računalu).

## 7.6 Završna pitanja i zadaci

Nakon aktivnog sudjelovanja i izvođenju vježbe u laboratoriju, potrebno je provjeriti je li vježba shvaćena u potpunosti. To ćete znati ako ste po završetku vježbe, u stanju samostalno i ispravno odgovoriti na završna pitanja (test u idućoj vježbi će se temeljiti na ovim pitanjima). Pored toga, poželjno je da samostalno pokušate produbiti i proširiti svoje znanje proučavajući navedene dodatne izvore. Također, pokušajte sami pronaći druge referentne izvore i materijale.

### 7.6.1 Završna pitanja u vezi izložene materije vježbe

- Što znači DBMS API?
- Što je ODBC i koje su mu prednosti?
- Koji problem ODBC rješava?
- Koje komponente obuhvaća sučelje ODBC-a?
- Nabrojite komponente arhitekture ODBC-a.
- Objasnite funkcije i ulogu aplikacije u ODBC arhitekturi.
- Objasnite funkcije i ulogu ODBC sučelja.
- Objasnite funkcije i ulogu upravitelja upravljačkih programa u ODBC arhitekturi.
- Objasnite funkcije i ulogu ODBC upravljačkih programa.
- Objasnite što predstavlja izvor podataka u ODBC arhitekturi.
- Što su razine usklađenosti?
- Kako se dijele razine usklađenosti?

### 7.6.2 Zadaci za dodatno samostalno produblјivanje znanja

- Pokušajte putem online izvora potražiti upute kako ostvariti ODBC konekciju Accessa s nekim drugim popularnim SUBP.

## 7.7 Literatura i dodatni izvori

1. [https://msdn.microsoft.com/en-us/library/ms710252\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/ms710252(v=vs.85).aspx)
2. [ODBC 2.0 Programmer's Manual](#)
3. [http://download.oracle.com/otn\\_hosted\\_doc/timesten/703/TimesTen-Documentation/ms.odbc.pdf](http://download.oracle.com/otn_hosted_doc/timesten/703/TimesTen-Documentation/ms.odbc.pdf)
4. <https://www.fbi-h-da.de/fileadmin/personal/h.erbs/Download/IntroductionToDatabases.pdf>

## Vježba 8:

## SQL DML

## 8.1 Motivacija

Dio SQL jezika koji se naziva DML – Data manipulation Language, sadrži implementaciju relacijskih operacija. Neke od tih operacija su implementirane eksplicitno, a neke implicitno. U ovoj ćemo vježbi obraditi SQL komande kojima je moguće rukovati entorkama u tablicama tako da ih pretražujemo ili transformiramo. Ovaj dio SQL jezika je najopsežniji i u praksi najviše korišten.

U ovoj vježbi su objašnjene komande potkrijepljene kratkim primjerima, a od studenata se očekuje da uz vlastitu imaginaciju isprobava različite slučajeve tih komandi na vlastitim primjerima ili na primjerima tablica iz prethodnih vježbi. U svakom slučaju, učenje ovih komandi traži prilično dodatnog vježbanja.

## Sadržaj vježbe:

VJEŽBA 8:	SQL DML	107
8.1	MOTIVACIJA	107
8.2	CILJEVI VJEŽBE – ISHODI UČENJA	108
8.3	SAMOSTALNA PRIPREMA VJEŽBE	108
8.4	TEORIJSKI DIO PRIPREME VJEŽBE	109
8.4.1	Komanda <i>SELECT</i> .....	109
8.4.2	Komanda <i>UNION</i> .....	113
8.4.3	Komanda <i>UPDATE</i> .....	114
8.4.4	Komanda <i>DELETE</i> .....	114
8.4.5	Komanda <i>INSERT INTO</i> .....	115
8.4.6	Komanda <i>SELECT INTO</i> .....	116
8.4.7	<i>SQL DML u Access-u</i> .....	117
8.5	SADRŽAJ I ZADATAK VJEŽBE	118
8.5.1	<i>Tijek izvođenja vježbe</i> .....	118
8.6	ZAVRŠNA PITANJA I ZADACI	120
8.6.1	<i>Završna pitanja u vezi izložene materije vježbe</i> .....	120
8.7	LITERATURA I DODATNI IZVORI	120

## 8.2 Ciljevi vježbe – ishodi učenja

Od studenata se očekuje da nakon uspješno završene vježbe:

- Shvate pojam i namjenu SQL –DML komandi.
- Razumiju i znaju sintaksu jednostavnijih DML komandi
- Nauče primjenjivati DML komande u Accessu.
- Razumiju kako pojedina DML komanda funkcionira.

## 8.3 Samostalna priprema vježbe

Obaveza studenata je pripremiti se za aktivno sudjelovanje u izvođenju vježbe:

- Proučite prethodnu materiju izloženu na predavanjima iz kolegija BP.
- Dobro proučite teorijski dio ove vježbe.
- Potražite dodatne informacije na internetu kojima možete produbiti potrebno znanje.
- Osim teorijske podloge za pripremu ove vježbe, pokušajte shvatiti i ostali dio materijala.
- Minimalno potrebna znanja za izradu vježbe:
  - Proučite i naučite sintaksu navedenih SQL komandi u dijelu 8.4:
    - SELECT – selekcija
    - UNION – unija
    - UPDATE – ažuriranje
    - DELETE – brisanje
    - INSERT – dodavanje

-



## 8.4 Teorijski dio pripreme vježbe

SQL DML, kao i DML, sadrži niz klauzula odnosno komandi. U ovoj vježbi ćemo obraditi DML komande i to s primjenom na Ms Access. DML komande se mogu odnositi na sljedeće akcije:

- SELECT – selekcija
- UNION – unija
- UPDATE – ažuriranje
- DELETE – brisanje
- INSERT – dodavanje

### 8.4.1 Komanda SELECT

Komanda SELECT obavlja skupovne operacije selekcije i projekcije te kao takva predstavlja jednu od važnijih i opširnijih SQL komandi. Ključna riječ SELECT obavlja operaciju projekcije, dok je za selekciju zadužena ključna riječ WHERE kao dio sintakse komande SELECT.

```
SELECT [predikat] { * | tablica.* | [tablica.]atribut1 [AS alias1] [, [tablica.]atribut2 [AS alias2] [, ...]] } FROM
tablični_izvor [, ...] [IN vanjska_baza] [WHERE... ] [GROUP BY... ]
[HAVING... ] [ORDER BY... ] [WITH OWNERACCESS OPTION]
```

Malim slovima pisane riječi imaju sljedeće značenje:

Riječ	Opis
<i>predicate</i>	Predikatom se defnirira postupanje s dupliciranim entorkama koje su dohvaćene select komandom. Moguće su sljedeće vrijednosti: ALL, DISTINCT, DISTINCTROW i TOP.
*	Oznaka koja selektira sva polja navedenih tablica nad kojima se provodi upit.
<i>table</i>	Naziv tablice ili više njih (odvajaju se zarezom) iz koje se selektiraju entorke prema zadanim kriterijima. U slučaju navođenja više tablica istih naziva, potrebno je korištenje kvalifikatora.
<i>field1, field2</i>	Nazivi atributa koji sadrže vrijednosti na kojima će se provoditi selekcija prema zadanim kriterijima.
<i>alias1, alias2</i>	Alternativni nazivi atributa koji će se prikazati umjesto onih iz tablica na kojima se primjenjuje komanda SELECT.
<i>tablični_izvor</i>	Naziv tablice ili više njih iz kojih se dohvaćaju podaci na temelju zadanih kriterija.
<i>vanjska_baza</i>	Naziv baze podataka koja sadrže neke od tablica navedenih kao izvor podataka (tableexpression) a ne nalaze se u trenutnoj bazi.

Predpodešena vrijednost predikata je ALL, a vraća sve entorke (uključujući duplikate) koje su prethodno zadovoljile selekcijske kriterije upita.

Primjeri:

```
SELECT ALL *  
FROM Djelatnik  
ORDER BY DjelatnikID;
```

```
SELECT *  
FROM Djelatnik  
ORDER BY DjelatnikID;
```

Predikat `DISTINCT` ispušta prikazati duplikate svih entorki s jednakim vrijednostima atributa i prikazuje samo prvu dohvaćenu entorku. U slučaju sljedećeg primjera, dohvatit će se sva jedinstvena prezimena svih zaposlenika iz tablice `Zaposlenik`. To znači, ukoliko u tablici `Zaposlenik` postoje radnici s jednakim prezimenima, prikazat će se prezime samo prvog radnika s tim prezimenom.

Primjer:

```
SELECT DISTINCT  
Prezime  
FROM Zaposlenik;
```

Predikat `TOP n [PERCENT]` omogućuje prikaz zadanog broja ili postotka entorki koji pripadaju gornjem ili donjem rasponu vrijednosti poredanih `ORDER BY` klauzulom. Izostankom klauzule `ORDER BY` prikazat će se proizvoljnih `n` entorki dohvaćenih komandom `SELECT`. Osim toga, u zadani broj ili postotak ne ulaze identične vrijednosti pa se tako raspon prikaza povećava za broj entorki s identičnim vrijednostima atributa na koje se predikat `TOP` primjenjuje.

Primjeri:

```
SELECT TOP 25  
Ime, Prezime  
FROM Student  
WHERE GodinaZavrsetka = 1994  
ORDER BY ProsječnaOcjena DESC;
```

```
SELECT TOP 10 PERCENT  
Ime, Prezime  
FROM Student  
WHERE GodinaZavrsetka = 1994  
ORDER BY ProsječnaOcjena ASC;
```

Predikat `DISTINCTROW` specifičan je za Access SQL, a koristi se samo kad se u argumentu klauzule `FROM` navodi više od jedne tablice. Za razliku od komande `DISTINCT` koja vraća prikaz entorki s različitim vrijednostima atributa iz kojih se podaci dohvaćaju, `DISTINCTROW` vraća prikaz vrijednosti iz međusobno različitim entorki. Ovaj predikat stoga ima najveću primjenu na nenormaliziranim ili nepovezanim tablicama koje ne sadrže jedinstvene entorke, što se rijetko viđa jer je protivno zahtjevima relacijske teorije.

Primjer:

```
SELECT DISTINCTROW Naziv  
FROM Artiki;
```

Iako je moguće da se u tablici Artiki nalazi više artikala istog naziva, kojima su međusobno različite neke od ostalih vrijednosti atributa koji ih opisuju, predikat DISTINCTROW će unatoč tome omogućiti prikaz svih naziva artikala jer se ti artikli upravo razlikuju bilo po vrijednosti primarnog ključa, bilo po nekoj drugoj vrijednosti atributa koji ih opisuju. Zato je korištenje ovog predikata rijetko učinkovito za ovakve jednostavne primjere jer se isti rezultat mogao dobiti i sljedećim izrazom:

```
SELECT Naziv FROM Artiki;
```

Smislenija namjena predikata DISTINCTROW prikazana je sljedećim primjerom, a detaljnije će biti obrađena u sljedećoj vježbi. U primjeru se koriste dvije povezane tablice Knjiga i Izdavač.

```
SELECT NazivIzdavača  
FROM Izdavač INNER JOIN Knjiga  
ON Izdavač.IDIzdavača=Knjige.Izdavač;
```

Kako je uobičajeno da jedan izdavač ima mnoštvo izdanih knjiga, tako će se kao rezultat ovog upita prikazati duplicirani nazivi izdavača i to onoliko puta zaredom koliko ima njihovih knjiga u tablici Knjige. U slučaju velikog broja knjiga pojedinih izdavača, rezultirajuća tablica postaje nepregledna te gubi osnovnu svrhu, a to je da nas informira koji sve izdavači imaju knjige u našoj knjižari. Dakle, da bismo dobili popis izdavača čije se knjige nalaze u našoj knjižari bez suvišnih ponavljanja njihovih naziva, potrebno je u komandu SELECT uključiti i predikat DISTINCT. Tada će rezultirajuća tablica naizgled biti točna u našem informiranju, no postoji slučaj kada to neće biti tako. Ukoliko se u tablici Izdavač pronađu izdavači koji dijele isti naziv iako su sasvim drugi pravni subjekti (imaju drugu adresu, sjedište itd.) tada se takvi neće pojaviti u rezultirajućoj tablici nego će se prikazati samo jedan naziv kao predstavnik oba izdavača. Da bi se izbjegla ovakva nepravilnost, potrebno je umjesto predikata DISTINCT koristiti predikat DISTINCTROW, koji osim naziva izdavača provjerava i ostale vrijednosti atributa pojedinih izdavača, te ih prikazuje u rezultirajućoj tablici ukoliko se razlikuju po barem jednoj vrijednosti atributa, što je zasigurno vrijednost primarnog ključa.

Predikat DISTINCTROW moguće je uključiti i kroz svojstva Access alata za upite postavljanjem vrijednosti svojstva Unique records na Yes. Također, predikat DISTINCT uključuje se u svojstvima upita postavljanjem vrijednosti svojstva Unique Values na Yes. Kao što je logično, istovremeno ne mogu biti aktivna oba svojstva.

Property Sheet ×

Selection type: Query Properties

General

Description	
Default View	Datasheet
Output All Fields	No
Top Values	All
Unique Values	No
Unique Records	Yes
Source Database	(current)
Source Connect Str	
Record Locks	No Locks
Recordset Type	Dynaset
ODBC Timeout	60
Filter	
Order By	
Max Records	
Orientation	Left-to-Right
Subdatasheet Name	
Link Child Fields	
Link Master Fields	
Subdatasheet Height	0cm
Subdatasheet Expanded	No
Filter On Load	No
Order By On Load	Yes

Klauzula FROM specificira tablice ili poglede na koje će komanda SELECT djelovati, tj. iz kojih će se dohvaćati entorke prema uspostavljenim kriterijima. Kao argument klauzule FROM (tzv. tablični\_izvor) moguće je navesti jednu ili više tablica međusobno odvojenih zarezima ili JOIN klauzulom o kojoj će više govora biti u sljedećim vježbama. Također, „tablični\_izvor“ segment komande SELECT može sadržavati i dio sintakse (AS alias) kojim se definiraju aliasi za tablice koje se navode u tom izrazu.

Primjer:

```
SELECT *
FROM Autor, Izdavač;
```

Ovakav iskaz komande SELECT generirat će kartezijev produkt dviju navedenih tablica Autor i Izdavač.

Klauzula WHERE specificira dio komande SELECT zadužene za operaciju selekcije entorki u prethodno specificiranim tablicama. Argument klauzule WHERE je izraz koji se može sastojati od naziva atributa, konstanti, aritmetičkih izraza (=, <, >, <=, >=, <>, BETWEEN), logičkih izraza (AND, OR, XOR, NOT) te raznih funkcija.

Primjeri:

```
WHERE Naslov LIKE "P*"
WHERE Len(Trim(Naslov)) > 10
WHERE Instr(Naslov, "Prohujalo") > 0 AND Len(Trim(Naslov)) > 10
WHERE Datum = #12/7/2002#
```

## 8.4.2 Komanda UNION

Operacijom unije moguće je kombinirati dva ili više pogleda, tablica u jedan ili međusobnih kombinacija. Sintaksa komande je sljedeća:

```
[TABLE] upit1 UNION [ALL] [TABLE] upit2 [UNION [ALL] [TABLE]
upit n [ ... ]]
```

Malim slovima pisane riječi imaju sljedeće značenje:

Riječ	Opis
<i>upit1...n</i>	SELECT komanda/naziv spremljenog pogleda/naziv tablice
<i>all</i>	Opcija kojom su u prikazu izvršenja komande pojavljuju sve entorke iz objekata koji sudjeluju u operaciji unije što uključuje mogućnost višestrukog prikaza istih entorki ukoliko takve postoje.

Primjeri:

```
TABLE [New Accounts] UNION ALL
SELECT *
FROM Klient
WHERE Kolicina > 1000;
```

```
TABLE Knjige
UNION ALL
SELECT * FROM NoveKnjige WHERE Cijena > 25.00
ORDER BY Naziv;
```

Za operaciju unije vrijede sljedeće zakonitosti:

- Svi pogledi i tablice koji sudjeluju u UNIJI moraju imati isti broj stupaca koji ne moraju biti istog podatkovnog tipa niti veličine.
- Stupci se u uniji uparaju prema redoslijedu iz svojih izvorišta.
- Aliasi se mogu koristiti u prvoj SELECT komandi (ukoliko postoji) da bi se definirali alternativni nazivi rezultirajućih atributa.
- ORDER BY klauzulu moguće je koristiti na kraju posljednjeg upita (upit n), a primjenjuju se na attribute iz prvog upita (upit 1).
- GROUP BY i HAVING klauzule mogu se koristiti u svrhu grupiranja podataka u svakom upitu iz komande UNION.
- Rezultirajuća tablica komande UNION ne može se mijenjati jer podaci iz njezinog prikaza ovise o objektima koje komanda UNION uključuje u svojoj definiciji.
- Komanda UNION nije dio SQL-92 standarda.

### 8.4.3 Komanda UPDATE

Komanda UPDATE upotrebljava se za promjenu vrijednosti atributa entorki zadane tablice na temelju postavljenih kriterija. Sintaksa komande UPDATE je sljedeća:

UPDATE *tablica* SET *nova\_vrijednost* WHERE *uvjet*;

Riječi pisane malim slovima imaju sljedeće značenje:

Riječ	Opis
<i>tablica</i>	Naziv tablice koja sadrži podatke čije vrijednosti želimo mijenjati.
<i>nova_vrijednost</i>	Izraz kojim se definira nova vrijednost atributa entorki određenih kriterijem.
<i>uvjet</i>	Izraz kojim se određuju entorke na koje će se primijeniti komanda UPDATE. Prethodi mu klauzula WHERE kojom se najavljuje kriterij.

Ovom komandom moguće je istovremeno mijenjati vrijednosti i u više od jedne entorke koje mogu pripadati različitim tablicama. Također, istovremeno je moguće mijenjati vrijednosti više atributa jedne ili više entorki iz jedne ili više tablica. Rezultat izvođenja ove komande ne rezultira novom tablicom već se sve promjene događaju nad tablicom ili tablicama pozvanim kroz argumente komande UPDATE. Zato je svaka promjena nepovratna.

Primjer:

```
UPDATE Narudžba
SET Količina = Količina * 1.1,
Volumen = Volumen * 1.03
WHERE Destinacija = 'IT';
```

### 8.4.4 Komanda DELETE

Komanda DELETE generira upit za nepovratno brisanje jedne ili više entorki iz jedne ili više tablica specificiranih klauzulom FROM koje zadovoljavaju kriterij specificiran klauzulom WHERE. Sintaksa komande DELETE je sljedeća:

DELETE [*tablica.\**] FROM *tablica* WHERE *uvjet*

Riječi pisane malim slovima imaju sljedeće značenje:

Riječ	Opis
<i>tablica</i>	Naziv tablice u kojoj se primjenjuje funkcija brisanja entorki.
<i>uvjet</i>	Izraz kojim se određuju entorke na koje će se primijeniti funkcija brisanja.

Ovom komandom je moguće obrisati sve entorke neke tablice pri čemu struktura tablice ostaje očuvana. Komanda DELETE može se primijeniti i na tablice u vezi „jedan prema više“ pri čemu se, ukoliko je omogućeno kaskadno brisanje, brišu sve povezane entorke.

## 8.4.5 Komanda INSERT INTO

Komandom INSERT INTO dodaju se nove entorke u tablici eksplicitnim navođenjem vrijednosti atributa novih entorki prema sljedećoj sintaksi:

Komanda za višestruko dodavanje entorki (kopiranje tablica):

```
INSERT INTO odredišni_objekt [(atribut1[, atribut2[, ...]])] [IN
vanjska_baza] SELECT [izvorišni_objekt.]atribut1[, atribut2[, ...]]
FROM tablični_izvor
```

Komanda za dodavanje pojedinačnih entorki:

```
INSERT INTO odredišni_objekt [(atribut1[, atribut2[, ...]])]
VALUES (vrijednost1[, vrijednost2[, ...]])
```

Riječi pisane malim slovima imaju sljedeće značenje:

Riječ	Opis
<i>odredišni_objekt</i>	Naziv tablice ili pogleda na koje se primjenjuje funkcija dodavanja entorki.
<i>field1, field2,...</i>	Nazivi atributa kojima se vrijednosti pridružuju (nakon definiranog odredišta – target) ili nazivi atributa iz kojih se podaci dohvaćaju ( nakon definiranog izvorišta – source).
<i>vanjska_baza</i>	Definicija vanjske baze podataka određene putanjom nakon klauzule IN.
<i>izvorišni_objekt</i>	Naziv tablice ili pogleda iz kojih se podaci koriste za popunjavanje vrijednosti atributa odredišne tablice.
<i>tablični_izvor</i>	Popis naziva tablica iz koje se kopiraju entorke u odredišnu tablicu. Ovaj argument može sadržavati naziv jedne tablice, spremljenog pogleda ili složenog SELECT upita.
<i>vrijednost1, vrijednost2</i>	Vrijednosti koje se eksplicitno pridružuju prethodno navedenim atributima nove entorki. Vrijednosti se međusobno odvajaju zarezima, a tekstualne vrijednosti omeđuju se znacima jednostrukih navoda ('tekst').

Primjeri:

```
INSERT INTO Knjiga
VALUES ("1-000-00000-0", "Uvod u telekomunikacijske mreže",1,125.00);
```

```
INSERT INTO Knjiga (ISBN,Naziv)
VALUES ("1-1111-1111-1", "Uvod u Access za početnike");
```

Ukoliko se ne navedu atributi odredišne tablice kojima se pridružuju navedene vrijednosti, potrebno je definirati vrijednosti svih atributa za novu entorku. Nedefinirane vrijednosti atributa automatski poprimaju vrijednost NULL.

Primjer:

```
INSERT INTO NoveKnjige
SELECT ISBN, Izdavač, Cijena
FROM Knjige
WHERE Cijena > 200;
```

## 8.4.6 Komanda SELECT INTO

Komandom SELECT INTO, specifičnom za Access SQL, kreira se nova tablica u bazi te popunjava podacima iz drugih tablica. Pri tome se ne prenose svojstva ograničenja iz postojećih tablica pa se ona moraju definirati naknadno. Sintaksa komande SELECT INTO je sljedeća:

```
SELECT atribut1[, atribut2[, ...]] INTO nova_tablica [IN  
vanjska_baza] FROM tablični_izvor
```

Riječi pisane malim slovima imaju sljedeće značenje:

Riječ	Opis
<i>atribut1,</i> <i>atribut2</i>	Naziv atributa iz kojih se podaci kopiraju u novu tablicu.
<i>nova_tablica</i>	Naziv nove tablice.
<i>vanjska_baza</i>	Definicija vanjske baze podataka određene putanjom nakon klauzule IN.
<i>tablični_izvor</i>	Naziv tablice ili pogleda iz kojih se podaci koriste za popunjavanje vrijednosti atributa nove tablice.

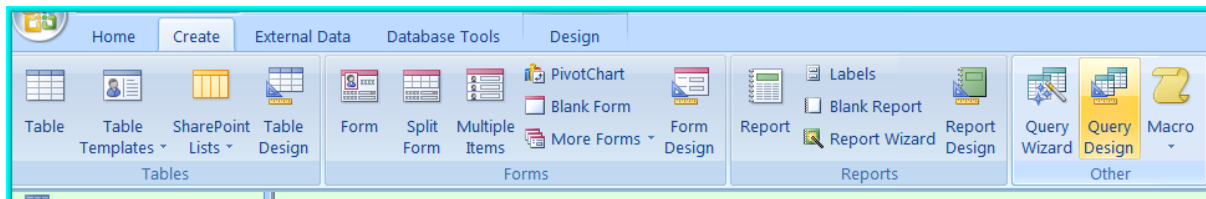
Primjer:

```
SELECT Naziv, ISBN  
INTO SkupeKnjige  
FROM Knjige  
WHERE Cijena > 450  
ORDER BY Naziv;
```



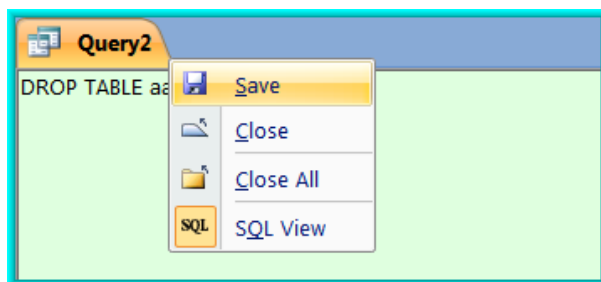
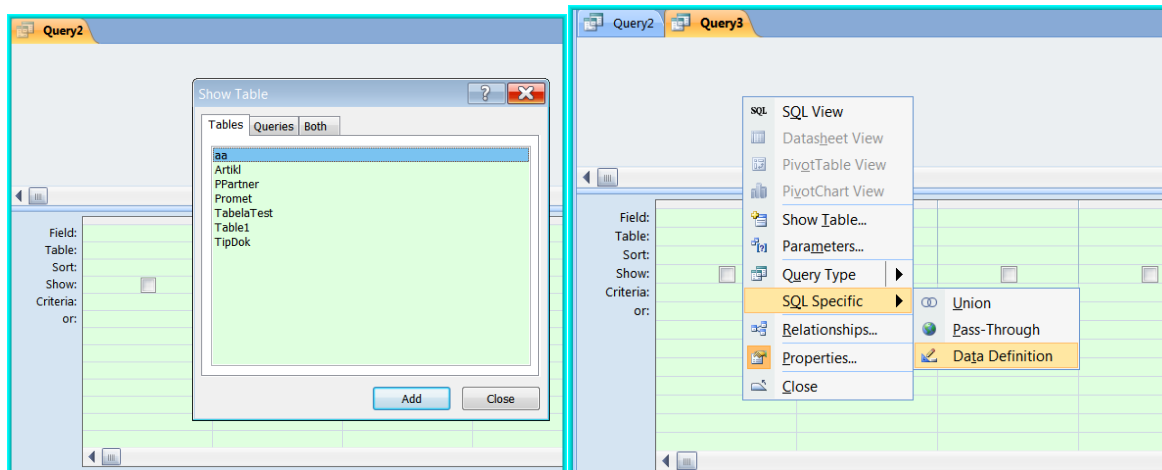
## 8.4.7 Kreiranje DML komandi u Access-u

U Accessu se SQL DML upiti mogu kreirati programskim putem ili putem Query Design alata:



Nakon što se otvori Query prozor potrebno je :

- Zatvoriti prozor Show Table
- Postaviti tip Query-ja na Data Definition (pritisnemo desnu tipku miša u prozoru Query)
- U postavljeni prozor upisati SQL DML komande
- Spremiti



## 8.5 Sadržaj i zadatak vježbe

U ovoj vježbi je potrebno otvoriti bazu **VJ-6-bp.accdb** te na osnovu nje definirati sve akcije u novoj bazi pomoću SQL DML naredbi

### 8.5.1 Tijek izvođenja vježbe

1. Otvorite bazu Vj-6-bp.accdb.
2. Pomoću Query-a upisivanjem SQL DML komande INSERT INTO dodajte po tri entorke u svaku tablicu.
3. Pomoću komande UPDATE povećajte cijenu svih artikala za 10%.
4. Pomoću komande UPDATE promijenite adresu poslovnog partnera pod šifrom 2. Nova adresa je Vukovarska 239a/II
5. Pomoću komande UPDATE promijenite datum upisa na 22.2.2013. međuskladišne primke broj 1 iz poslovne godine 2013.
6. Kopirajte sav sadržaj tablice PPartner u tablicu PPartnerNovo koristeći komandu SELECT INTO.
7. U tablicu Radnik dodajte novog radnika: Marko Marković, Vinkovačka 33, Virovitica.
8. Obrišite sve radnike iz tablice Radnik kojima prezime završava slogom **vić**.
9. Kreirajte pogled kojim ćete prikazati sve entorke tablice Artikal te ga spremite pod nazivom Pogled9.
10. Kreirajte pogled kojim ćete prikazati sve entorke tablice Artikal kojima je cijena između 10 i 50 kn te ih poredajte prema cijeni od najniže prema najvišoj.
11. Kreirajte pogled kojim ćete prikazati sve prometne dokumente kojima je ulazna količina između 10 i 50 te kojima je izlazna količina manja od 40.
12. Kreirajte pogled kojim ćete prikazati naziv, cijenu i šifru dobavljača prvih 3 proizvoda s najmanjom cijenom.
13. Kreirajte pogled kojim ćete prikazati naziv, cijenu i šifru dobavljača posljednjih 3 proizvoda s najmanjom cijenom.
14. Kreirajte pogled kojim ćete prikazati naziv, cijenu i šifru dobavljača prvih 3 proizvoda s najvećom cijenom.
15. Kreirajte pogled kojim ćete prikazati prvih 50% artikla iz pogleda Pogled9.
16. U bazi Vj-6-2013 kreirajte tablicu Profesor sa sljedećim atributima: JMBG, Ime, Prezime, email. Kreirajte tablicu Student sa sljedećim atributima: JMBG, Ime, Prezime, email, Profesor. Povežite navedene tablice odgovarajućom definicijom stranog ključa.

17. Pomoću SQL komande INSERT INTO dodajte sljedeće entorke u tablice Student i Profesor:

Profesor		Student		
	JMBG	Ime	Prezime	email
+	12332545324	Jozip	Kristić	josip.kristic@vsmti.hr
+	45436564352	Mirta	Pantek	mirta.pantek@vsmti.hr
+	57634534535	Antun	Kolar	antun.kolar@vsmti.hr
+	67545356435	Petra	Lukić	petra.lukic@vsmti.hr

Profesor		Student			
	JMBG	Ime	Prezime	email	Profesor
	65445656464	Ivo	Bujanić	ivo.bujanic@vsmti.hr	45436564352
	22256547746	Karla	Smidth	karla.smidth@vsmti.hr	45436564352
	44455242534	Ivona	Mirić	ivona.miric@vsmti.hr	67545356435
	56523542242	Mateja	Horvat	mateja.horvat@vsmti.hr	12332545324
	76634124234	Ivan	Begović	ivan.begovic@vsmti.hr	12332545324

18. Pomoću komande UNION kreirajte mailing listu koja sadrži imena, prezimena i e-mail adrese svih studenata i profesora. Spremite uniju pod nazivom Unija\_ProfesorStudent.
19. Na temelju unija Unija\_ProfesorStudent kreirajte pogled kojim ćete postići sljedeći prikaz. Obratite pozornost na nazive atributa (koristite SQL klauzulu za definiranje aliasa).

## 8.6 Završna pitanja i zadaci

Nakon aktivnog sudjelovanja i izvođenju vježbe u laboratoriju, potrebno je provjeriti je li vježba shvaćena u potpunosti. To će te znati ako ste po završetku vježbe, u stanju samostalno i ispravno odgovoriti na završna pitanja (test u idućoj vježbi će se temeljiti na ovim pitanjima). Pored toga, poželjno je da samostalno pokušate produbiti i proširiti svoje znanje proučavajući navedene dodatne izvore. Također, pokušajte sami pronaći druge referentne izvore i materijale.

### 8.6.1 Završna pitanja u vezi izložene materije vježbe

- Što znači DML?
- Koje su osnovne komande DML-a?
- Što znači klauzula SELECT?
- Što znači klauzula FROM u komandi SELECT? Je li ona obavezna i zašto?
- Što znači klauzula WHERE u komandi SELECT? Je li ona obavezna i zašto?
- Što znači klauzula ORDER BY? Je li ona obavezna i zašto?
- Što znači klauzula SELECT INTO?
- Što znači klauzula INSERT INTO?
- Što znači klauzula UNION?
- Što znači klauzula DELETE?
- Što znači klauzula TOP?
- Kako se u Access Query-u definira DML query?

## 8.7 Literatura i dodatni izvori

- 1) <http://sql.org/sql-database/sql-tutorial/>
- 2) <http://www.w3schools.com/sql/>
- 3) [Data Manipulation Language \(https://msdn.microsoft.com/en-us/library/office/dn124073.aspx\)](https://msdn.microsoft.com/en-us/library/office/dn124073.aspx)

**Vježba 9:****SQL spojevi**

## 9.1 Motivacija

Spoj je jedna od složenijih operacija u relacijskoj teoriji podataka. U SQL-u se spoj zbog dozvoljene pojave duplikata ponaša nešto drugačije nego u teoriji. Spoj je u SQLu dio SELECT komande koji se definira klauzulom JOIN u FROM dijelu. Važnost spoja je u Select komandi velika. Ovoj vježbi treba posvetiti osobitu pažnju.

Ovom vježbom želimo studentima predstaviti pojam spojeva u SQL-u, odnosno u dio SQL komandi koje se odnose na spajanje dviju ili više tablica.

### Sadržaj vježbe:

<b>VJEŽBA 9:</b>	<b>UVOD U MICROSOFT ACCESS</b>	<b>  121</b>
9.1	MOTIVACIJA	121
9.2	CILJEVI VJEŽBE – ISHODI UČENJA	122
9.3	SAMOSTALNA PRIPREMA VJEŽBE	122
9.4	TEORIJSKI DIO PRIPREME VJEŽBE	123
9.4.1	SQL spojevi .....	123
9.4.2	Unutarnji spoj (INNER JOIN) .....	123
9.4.3	Vanjski spoj (OUTER JOIN) .....	124
9.5	SADRŽAJ I ZADATAK VJEŽBE	125
9.5.1	Tijek izvođenja vježbe .....	125
9.6	ZAVRŠNA PITANJA I ZADACI	126
9.6.1	Završna pitanja u vezi izložene materije vježbe .....	126
9.7	LITERATURA I DODATNI IZVORI	126

## 9.2 Ciljevi vježbe – ishodi učenja

Od studenata se očekuje da nakon uspješno završene vježbe:

- Shvate pojam i namjenu JOIN komandi.
- Nauče primjenjivati JOIN komande u Accessu.
- Razumiju posljedice primjene JOIN komandi.

## 9.3 Samostalna priprema vježbe

Obaveza studenata je pripremiti se za aktivno sudjelovanje u izvođenju vježbe:

- Proučite prethodnu materiju izloženu na predavanjima iz kolegija BP.
- Dobro proučite teorijski dio ove vježbe.
- Potražite dodatne informacije na internetu kojima možete produbiti potrebno znanje.
- Osim teorijske podloge za pripremu ove vježbe, pokušajte shvatiti i ostali dio materijala.
- Minimalno potrebna znanja za pristup vježbi:
  - Dobro proučite i naučite značenje i sintaksu spoja u SQL-u

## 9.4 Teorijski dio pripreme vježbe

### 9.4.1 SQL spojevi

Do sada smo bezuvjetno spajali tablice kako bismo si omogućili jednostavniju ili opsežniju manipulaciju podacima iz više tablica. Pri tome je rezultirajući pogled odgovarao Kartezijevom produktu tablica uključenih u bezuvjetnom spoju.

Primjer:

```
SELECT * FROM Tablica1, Tablica2;
```

Češći je slučaj spajanja tablica prema nekom kriteriju, poput jednakosti.

Primjer:

```
SELECT Artikl.IdArtikla, Artikl.Naziv, Artikl.Cijena, PPartner.Naziv,
PPartner.Adresa AS [Adresa Dobavljača]
FROM Artikl, PPartner
WHERE Artikl.Dobavljač = PPartner.IdPP;
```

U ovom primjeru spojene su tablice Artikl i PPartner prema uvjetu jednakosti šifre dobavljača.

Kombiniranje entorki (spajanje) iz dvije ili više tablica prema vrijednostima nekog zajedničkog atributa moguće je izvršiti putem spoja. Prema načinu spajanja, spojevi se dijele na unutarnji (INNER JOIN) i vanjski (OUTER JOIN) koji se dijeli na lijevi (LEFT JOIN) i desni (RIGHT JOIN). Klauzula JOIN ne može postojati kao samostalan SQL izraz te ju je potrebno iskoristiti kao dio nekog drugog, šireg izraza, poput SELECT.

### 9.4.2 Unutarnji spoj (INNER JOIN)

Unutarnji spojevi služe za spajanje entorki iz više tablica u jedinstveni skup vrijednosti atributa na osnovi zadovoljenja uvjeta postavljenog na vrijednosti iz jednog ili više zajedničkih atributa.

Sintaksa unutarnjeg spoja je sljedeća:

```
FROM tablica1 INNER JOIN tablica2 ON tablica1.atribut1 operator
table2.atribut1 [{AND|OR ON tablica1.atribut2 operator
tablica2.atribut2}, ...]
```

Riječ	Opis
<i>tablica1</i> , <i>tablica2</i>	Nazivi tablica koje se spajaju.
<i>atribut1</i> , <i>atribut2</i>	Nazivi atributa na temelju kojih se izvodi spajanje. Ovi atributi moraju biti istog podatkovnog tipa.
<i>operator</i>	Operator usporedbe (=, <, >, <=, >=, <>)

Primjer:

```
SELECT * from Artikl INNER JOIN PPartner ON Artikl.Dobavljač =
PPartner.IdPP;
```

U primjeru će se napraviti spoj vrijednosti atributa tablice Artikl s vrijednostima atributa tablice PPartner za sve entorke iz tablice Artikl kojima se vrijednosti atributa Dobavljač podudara s vrijednosti atributa IdPP u tablici PPartner. Drugim riječima, proširujemo prikaz tablice Artikl s podacima o Dobavljaču za svaku entorku iz Tablice Artikl. Pri tome ne kreiramo novu tablicu, nego upit .

### 9.4.3 Vanjski spoj (OUTER JOIN)

Postoje slučajevi kada unutarnji spoj nije adekvatan za željeni prikaz podataka. U slučaju potrebe za spajanjem tablica na temelju vrijednosti jednog ili više zajedničkih atributa, bez obzira postoje li te vrijednosti u nekoj od tablica, koristit će se vanjski spojevi.

Općenita sintaksa vanjskog spoja je sljedeća:

```
FROM tablica1 [ LEFT | RIGHT ] JOIN tablica2 ON
tablica1.atribut1 operator tablica2.atribut2 [{AND|OR ON
tablica1.atribut2 operator tablica2.atribut2}, ...]
```

U slučaju lijevog vanjskog spoja koristit će se klauzula LEFT JOIN, a za desni vanjski spoj klauzula RIGHT JOIN. Lijevi vanjski spoj kombinira sve entorke prve (lijeve) tablice iako ne postoje odgovarajuće entorke druge (desne) tablice.

Primjer:

```
SELECT * FROM PPartner LEFT JOIN Artikl ON
PPartner.IdPP=Artikl.Dobavljač;
```

U ovom primjeru će se prikazati sve entorke tablice PPartner čak iako ne postoje artikli vezani za neke od artikala u tablici Artikl. Za one dobavljače koji postoje u tablici PPartner, a nemaju niti jedan artikl u tablici Artikl, prikazat će se NULL vrijednosti za sve vrijednosti atributa iz tablice Artikl.

Desni vanjski spoj kombinira sve entorke druge (desne) tablice i u slučaju da ne postoje odgovarajuće entorke prve (lijeve) tablice.

Primjer:

```
SELECT * FROM Artikl RIGHT JOIN PPartner ON
PPartner.IdPP=Artikl.Dobavljač;
```

Ovaj primjer djeluje suprotno prošlom primjeru pa se prikazuju sve entorke desne tablice (tablica PPartner) bez obzira postoje li artikli za svakog od dobavljača iz tablice PPartner. Za one dobavljače koji nemaju odgovarajuće artikle, prikazat će se NULL vrijednosti na mjestima atributa tablice Artikl.



## 9.5 Sadržaj i zadatak vježbe

### 9.5.1 Tijek izvođenja vježbe

1. Otvorite bazu podataka priloženu vježbi 5.
2. Kreirajte unutarnji spoj tablica Artikl i PPartner te spremite upit pod nazivom Query2.
3. Kreirajte lijevi vanjski spoj tablica Artikl i PPartner te spremite upit pod nazivom Query3.
4. Kreirajte desni vanjski spoj tablica PPartner i Artikl te spremite upit pod nazivom Query4.
5. Analizirajte razlike između spojeva u zadacima 2, 3 i 4.
6. Otvorite testnu bazu podataka NWT priloženu 1. vježbi i riješite sljedeće zadatke.
7. Kreirajte upit kojim ćete saznati naziv proizvoda čiji je ID=12.
8. Kreirajte upit kojim ćete prikazati ID klijenta, naziv poduzeća, Ime i Prezime, zanimanje (Job Title) za sve klijente iz savezne države Washington (WA).
9. Kreirajte upit kojim ćete prikazati naziv (Product Name), zalihe (Reorder Level), kategoriju (Category) te osnovnu cijenu (Standard Cost) napitaka (kategorija Beverages) koje imaju barem 10 jedinica u zalihama ili kojima je jedinična cijena veća od 10kn.
10. Kreirajte upit kojim ćete prikazati ID proizvoda (ID), naziv proizvoda (Product Name) i osnovnu cijenu proizvoda (Standard Cost) za sve proizvode kojima je jedinična cijena između 10 i 15kn.
11. Kreirajte upit kojim ćete prikazati ID narudžbe (Order ID), ID klijenta (Customer ID), datum narudžbe (OrderDate) te ime i prezime prodajnog predstavnika (FirstName, LastName) koji je izradio narudžbe u periodu između 6.3.2006. i 22.4.2006.
12. Kreirajte upit kojim ćete prikazati ID narudžbe (Order ID) i datum narudžbe (Order Date) za sve proizvode kojima se u nazivu nalazi segment „Traders Dried“.
13. Kreirajte upit kojim ćete prikazati ime i prezime zaposlenika te ID narudžbe i datum narudžbe za narudžbe koje je kreirao dotični zaposlenik za razdoblje od 6.3.2006. do 1.7.2006.
14. Kreirajte upit kojim ćete prikazati ime i prezime zaposlenika (First Name, Last Name) te ID narudžbe (Order ID) i datum narudžbe (Order Date) za sve zaposlenike koji nisu imali narudžbe u razdoblju od 1.1.2006. do 30.7.2006.

## 9.6 Završna pitanja i zadaci

Nakon aktivnog sudjelovanja i izvođenja vježbe u laboratoriju, potrebno je provjeriti je li vježba shvaćena u potpunosti. To ćete znati ako ste po završetku vježbe, u stanju samostalno i ispravno odgovoriti na završna pitanja (test u idućoj vježbi će se temeljiti na ovim pitanjima). Pored toga, poželjno je da samostalno pokušate produbiti i proširiti svoje znanje proučavajući navedene dodatne izvore. Također, pokušajte sami pronaći druge referentne izvore i materijale.

### 9.6.1 Završna pitanja u vezi izložene materije vježbe

- Što su spojevi u SQL-u i kako se ostvaruju?
- U koje komande se spoj neposredno ugrađuje?
- Koji operatori usporedbe mogu biti uključeni u sintaksi SQL spojeva?
- Kako se primjena pojedinog operatora odražava na rezultat spoja?
- Može li u spoju sudjelovati samo jedna tablica?
- Može li u spoju sudjelovati više od dvije tablice?
- Što je to lijevi spoj?
- Što je to desni spoj?
- Što je to unutrašnji spoj?
- U okviru koje klauzule se tipično koristi spoj?
- 

## 9.7 Literatura i dodatni izvori

- 1) [Perform Joins Using Access SQL \(https://msdn.microsoft.com/en-us/library/bb243855\(v=office.12\).aspx\)](https://msdn.microsoft.com/en-us/library/bb243855(v=office.12).aspx)
- 2) <http://sql.org/sql-database/sql-tutorial/>
- 3) <http://www.w3schools.com/sql/>
- 4) [Join tables and queries \(https://support.office.com/en-ca/article/Join-tables-and-queries-3f5838bd-24a0-4832-9bc1-07061a1478f6\)](https://support.office.com/en-ca/article/Join-tables-and-queries-3f5838bd-24a0-4832-9bc1-07061a1478f6)

## Vježba 10: Firebird – instalacija i upotreba

### 10.1 Motivacija

U ovoj se vježbi studente uvodi u FirebirdSQL - jedan široko rasprostranjen sustav za upravljanje relacijskim bazama podataka, oko kojega je razvijena čvrsta baza stručne i šire zajednice s vrlo dobrom podrškom za sve nove korisnike u vidu dostupnosti referentne literature te foramskih diskusijskih lista. FirebirdSQL će biti predstavljen kroz kratki vodič od početnih koraka instalacije do jednostavnog rada s FlameRobin grafičkim alatom.

Nakon instalacije, trebat će izraditi nekoliko SQL primjera.

#### Sadržaj vježbe :

<b>VJEŽBA 10: FIREBIRD SUBP</b>	<b>127</b>
10.1 MOTIVACIJA	127
10.2 CILJEVI VJEŽBE – ISHODI UČENJA	128
10.3 SAMOSTALNA PRIPREMA VJEŽBE	128
10.4 TEORIJSKI DIO PRIPREME VJEŽBE	129
10.4.1 Uvod u Firebird RDBMS .....	129
10.4.2 Firebird instalacija .....	130
10.4.3 Vizualni alati za rad s Firebird relacijskom bazom podataka .....	131
10.4.4 Firebird ODBC driver .....	134
10.5 SADRŽAJ I ZADATAK VJEŽBE	135
10.5.1 Tijek izvođenja vježbe .....	135
10.6 ZAVRŠNA PITANJA I ZADACI	136
10.7 LITERATURA I DODATNI IZVORI	136

## 10.2 Ciljevi vježbe – ishodi učenja

Od studenata se očekuje da nakon uspješno završene vježbe:

- Upoznaju s Firebird RDBMS-om te grafičkim alatom Flame Robin.
- Razumiju osnovne razlike u odnosu na druge proizvode istog tipa.
- Znaju samostalno provesti instalaciju i konfiguraciju Firebird RDBMS-a te Flame Robin vizualnog upravljačkog okruženja.
- Upoznaju se s osnovnim dijalektalnim razlikama u odnosu na Access SQL.
- U stanju su kreirati bazu podataka na temelju zadanih parametara.
- Znaju kreirati i izvršavati osnovne upite u skladu s trenutnom razinom poznavanja SQL-a.

## 10.3 Samostalna priprema vježbe

Obaveza studenata je pripremiti se za aktivno sudjelovanje u izvođenju vježbe:

- Ponoviti prethodno izložene teme iz osnova SQL jezika.
- Dobro proučite teorijski dio ove vježbe.
- Potražite dodatne informacije na internetu kojima možete produbiti potrebno znanje.
- Osim teorijske podloge za pripremu ove vježbe, pokušajte shvatiti i ostali dio materijala.
- Ako su vam neki dijelovi materijala eventualno nerazumljivi pripremite pismena pitanja.
- Prije pristupanja vježbi, minimalno biste trebali znati odgovore na pitanja:
  - Što RDBMS? Što je Firebird?
  - Koje su karakteristike Firebird RDBMS-a?
  - Što je FlameRobin? Postoji li alternativa navedenom alatu i možete li navesti neke od njih? Poslužite se pritom internetskim resursima.

## 10.4 Teorijski dio pripreme vježbe

### 10.4.1 Uvod u Firebird RDBMS

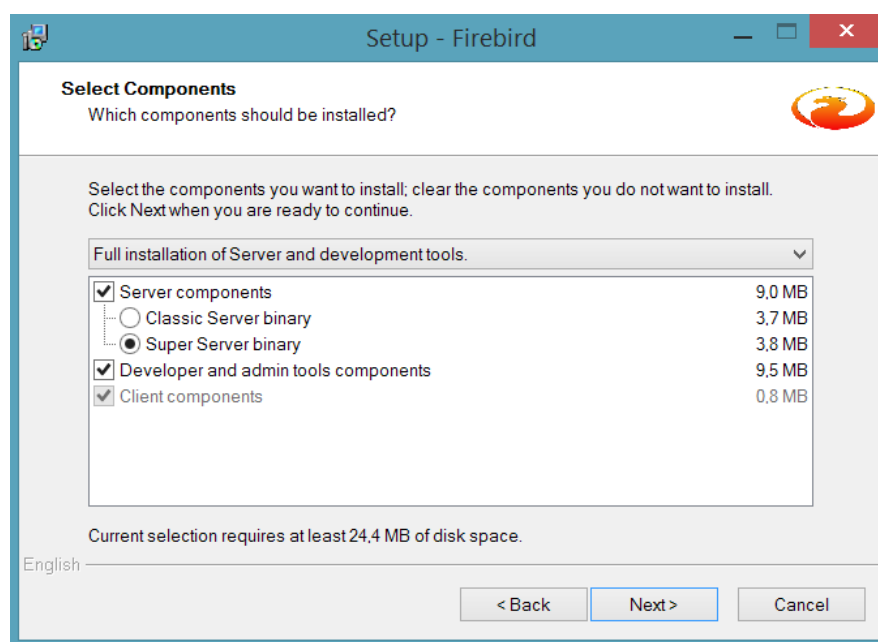
Firebird, poznat i kao FirebirdSQL, prepoznatljivo je ime na današnjem tržištu otvorenih i besplatnih sustava za upravljanje relacijskim bazama podataka (SUBP). Nastao je na temelju Borlandovog InterBase 6.0 izvornog koda i danas je dostupan pod otvorenom licencom te je besplatan za preuzimanje i instalaciju te implementaciju u bilo koje svrhe, što ga između ostalih bitnih prednosti čini ozbiljnim konkurentom sličnim poznatijim komercijalnim sustavima zatvorenog koda. Današnji Firebird obuhvaća veći dio ANSI SQL-92 i SQL-2003 standarda te i po tome prati korak s poznatijim proizvodima iste namjene. Prvo izdanje Firebirda seže u 1984. godinu te se od tada do danas razvio u ozbiljan SUBP vrlo velikih mogućnosti, a kao bitne prednosti ističu se:

- omogućava rad s bazama podataka od nekoliko kilobajta do onih reda veličine stotine gigabajta uz minimalnu administraciju
- potpuna podrška za spremljene procedure i trigere
- transakcije sukladne ACID zahtjevima
- vrlo malo zauzeće instalacijom, tzv. footprint
- omogućuje definiranje svih svojstava referencijalnog integriteta
- podržava MGA (Multi Generation Architecture) tehnologiju koja u prijevodu znači „čitači ne blokiraju pisalice i obrnuto“. Osnovna ideja je u čuvanju višestrukih verzijama entiteta u bazi sve dok su potrebne u aktivnim transakcijama.
- podrška za eksterne funkcije (UDF – user defined functions) koje mogu biti pisane proizvoljnim programskim jezikom
- vrlo jednostavna instalacija gotovo bez potrebe za složenim podešavanjima
- vrlo dobra podrška stručne i korisničke zajednice od kojih neki pripadaju i Firebird zakladi (neprofitnoj organizaciji oko koje su okupljeni razni stručnjaci, tvrtke i entuzijasti s ciljem kontinuiranog razvoja Firebird SUBP-a i pratećih proizvoda)
- velika podrška dodatnih aplikacija koje olakšavaju rad s Firebird RDBMS-om i proširuju njegovu funkcionalnost (razne GUI aplikacije, administrativni alati, alati za replikaciju i sl.)
- vrlo dobra povezivost s ostalim aplikacijama i SUBP sustavima (dbExpress drivers, ODBC, OLEDB, .Net provider, JDBC native type 4 driver, Python module, PHP, Perl i td.)
- podrška za većinu poznatih operativnih sustava (Windows, Linux, Solaris, MacOS, HP-UX and FreeBSD i td.)
- podrška za inkrementalni backup koji se odvija u manjim koracima tijekom određenog vremenskog perioda pri čemu se pohranjuju podaci koji su se promijenili od zadnje promjene
- podrška za 64-bitne sustave

- potpuna implementacija kursora u PSQL-u
- info-tablice za nadzor raznih parametara aktivnosti i trenutnog stanja baze
- podrška za privremene tablice (brišu se krajem klijentske sesije)
- Audit and Trace Services API za nadzor prošlih akcija s bazom u svrhu otklanjanja problema u radu s bazom nastalih prije nekog vremena, kao i izvlačenja raznih statistika o bazi

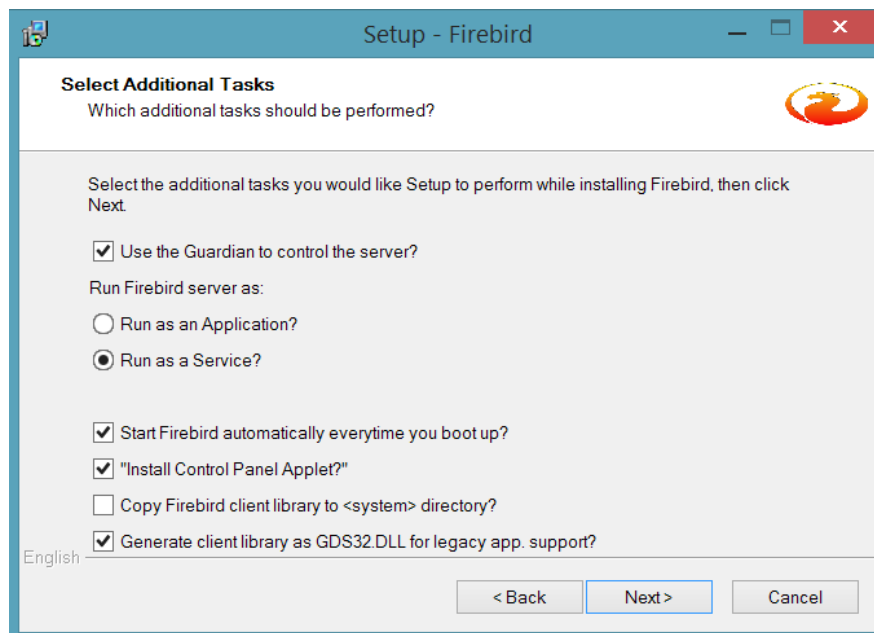
## 10.4.2 Firebird instalacija

Budući je instalacijska datoteka Firebirda veličine svega 7MB, instalacija je vrlo brza i nezahtjevna po pitanju drugih postavki. Firebird instalacijsku datoteku moguće je preuzeti za odgovarajući operativni sustav na službenim stranicama Firebird udruženja (<http://www.firebirdsql.org/en/downloads/>). Pri instalaciji se otkrivaju dvije arhitekture Firebird servera: *SuperServer* i *Classic Server*. Preporuka je za Windows platforme instalaciju započeti SuperServer inačicom. SuperServer radi kao jedan proces koji osluškuje klijentske konekcije te dijeli svoj međuspremnik s metapodacima među konekcijama na bazu te koristi niti (threads) za upravljanje konekcijama. *Classic* inačica za svaku konekciju pokreće individualan poslužiteljski proces, dok *SuperClassic* nudi višenitnu jednoprocesnu serversku inačicu s nezavisnim međuspremnikom za svaku konekciju. *Super Server* arhitektura općenito pruža bolje performanse za OLTP (*On-Line Transaction Processing*) sustave u kojima se očekuju brze i učestale transakcije (INSERT, UPDATE i DELETE) te primjena jednostavnijih upita. *Classic Server* arhitektura pogodnija je za OLAP (*On-Line Analytical Processing*) sustave koje karakterizira mali broj transakcija te primjena složenijih, analitički orijentiranih agregiranih upita. Za rad s relacijskim bazama podataka na razini ovoga kolegija preporuka je stoga odabrati *Super Server* arhitekturu kao na sljedećoj slici.



Nakon odabrane željene arhitekture u postupku instalacije pojavljuje se prozor gdje se odabiru dodatne specifičnosti odabrane inačice poslužitelja. Jedan od ponuđenih izbora je korištenje *Guardian*

uslužnog programa koji nadzire rad poslužiteljskog procesa te se brine za njegovo brzo uspostavljanje nakon nepredviđenih rušenja.

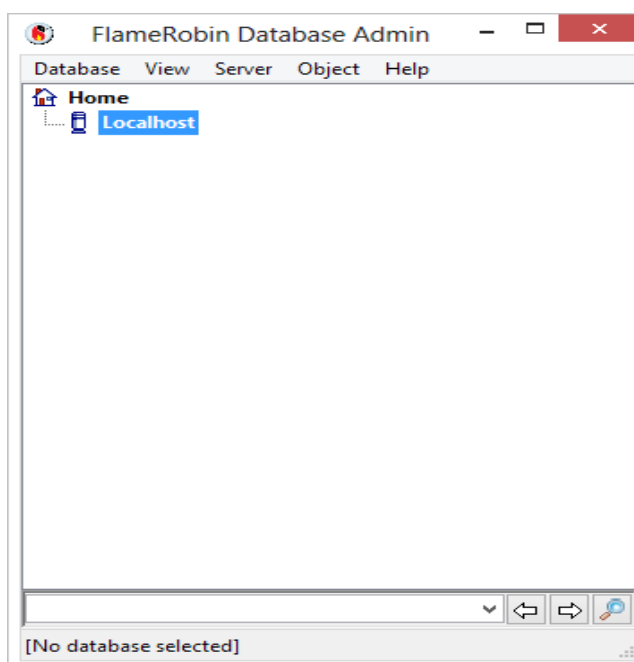


Nakon instalacije Firebird RDBMS-a automatski je kreiran i administratorki račun **SYSDBA** s početnom lozinkom **masterkey**, koju je kasnije poželjno promijeniti radi sigurnosti.

### 10.4.3 Vizualni alati za rad s Firebird relacijskom bazom podataka

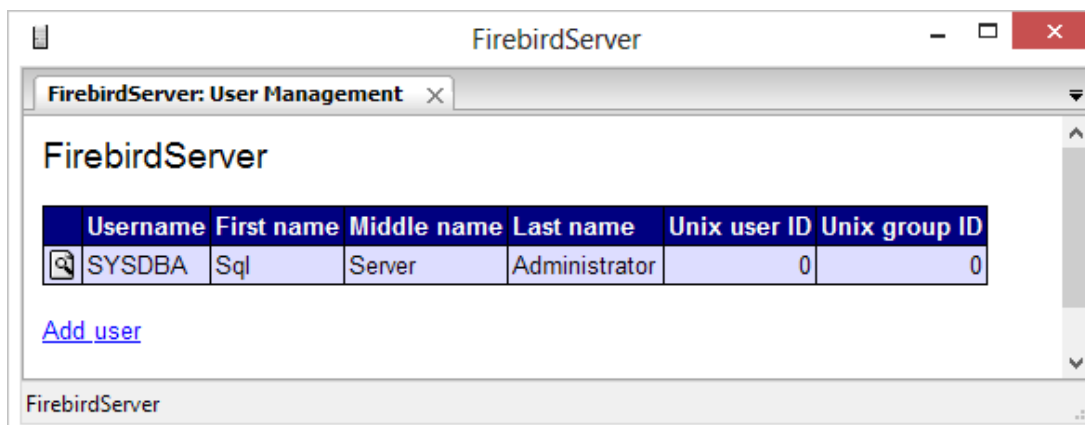
Jedan od zastupljenijih administratorskih alata za rad s Firebird SUBP-om poznat je pod nazivom **FlameRobin**, dostupan pod otvorenom licencom na sljedećoj lokaciji:

<http://sourceforge.net/projects/flamerobin/files>



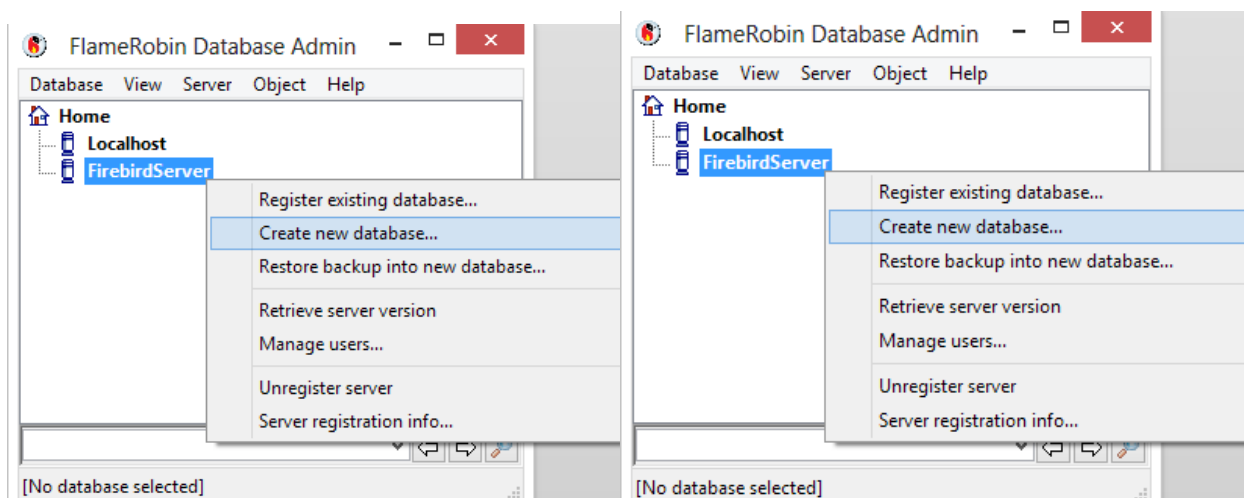
FlameRobin administratorsko sučelje prema Firebird RDBMS-u

Do upravljanja Firebird SUBP korisnicima u FlameRobin GUI-u dolazi se kroz izborničku kategoriju Server u traci izbornika, klikom na opciju Manage users. Za inicijalni ulazak u tablicu korisnika potrebno je proći autorizaciju pristupa s administratorskim računom (***SYSDBA, masterkey***) gdje se potom može promijeniti predefinirana lozinka SYSDBA korisnika te kreirati novi korisnici.



Prije kreiranja nove baze podataka potrebno je definirati instancu Firebird poslužitelja, što se postiže desnim klikom na Home te odabirom opcije *Register server...* Nakon toga potrebno je definirati naziv poslužitelja (*Display name*), naziv lokalnog računala (*Hostname*) te broj TCP/IP porta (*Port number*) na kojemu će poslužitelj slušati konekcije. Uobičajeni port za Firebird uslugu je 3050, a naziv poslužitelja je proizvoljan. Prvom instalacijom Firebird SUBP-a generirana je predefinirana instanca poslužitelja te je moguće kreirati baze podataka i pod tom instancom.

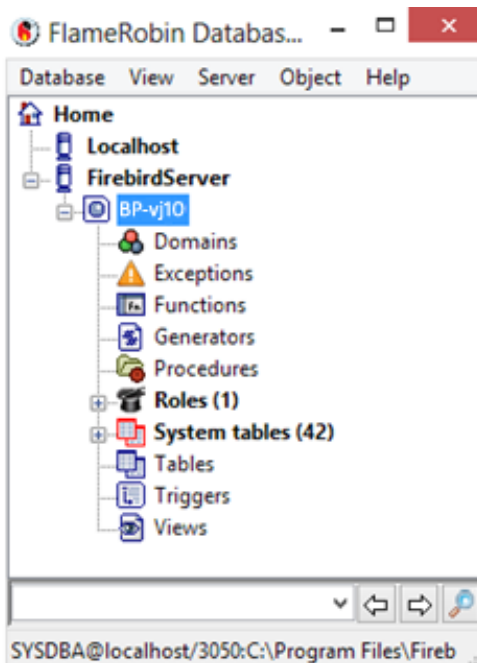
Desnim klikom miša na naziv lokalnog računala, koji predstavlja Firebird Superserver, otvara se izbornik u kojemu je moguće pronaći opciju *Create New Database* (Izradi novu bazu podataka). Klikom na navedenu opciju otvara se dijaloški okvir u kojemu se definiraju svojstva nove baze: naziv (*Display Name*), kodna shema (*Charset*) te korisničko ime i lozinka administratora baze.



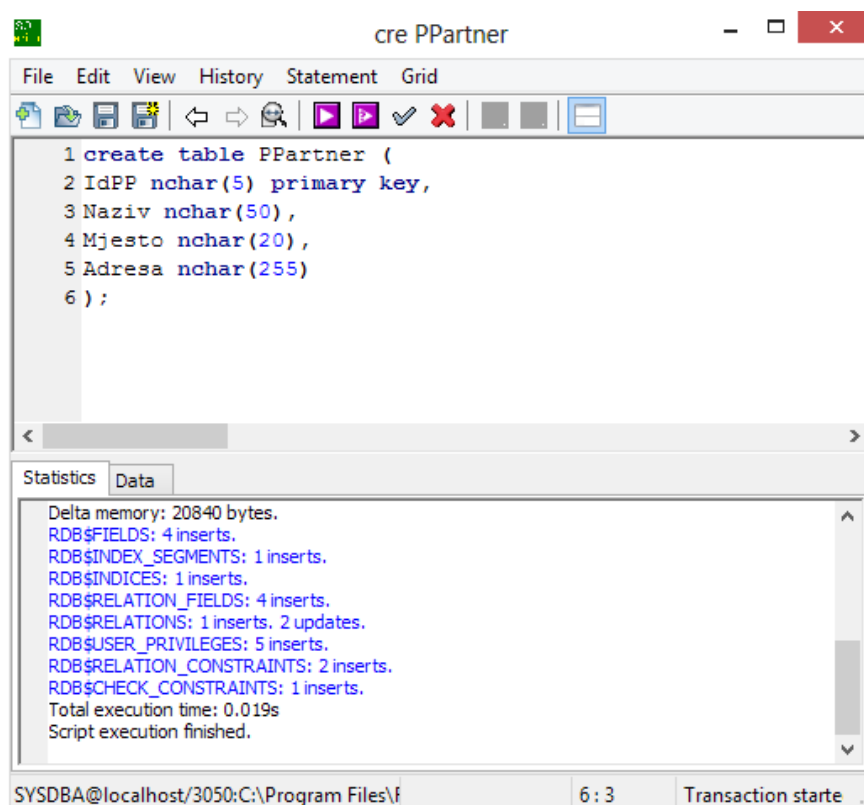
Nakon pritiska na dugme *Create* stvorit će se prazna baza podataka pod nazivom BP-vj10, kao što je prethodno definirano.



Dvoklikom na novu ikonu, s nazivom nove baze podataka, ispod instance FirebirdServera pokreće se spajanje FlameRobin sučelja na Firebird bazu podataka BP-vj10. Nakon uspješnog spajanja na odabranu bazu pojavljuju se pod njezinim nazivom svi pripadajući elementi: domene, funkcije, procedure, korisnici, tablice, pogledi i dr.



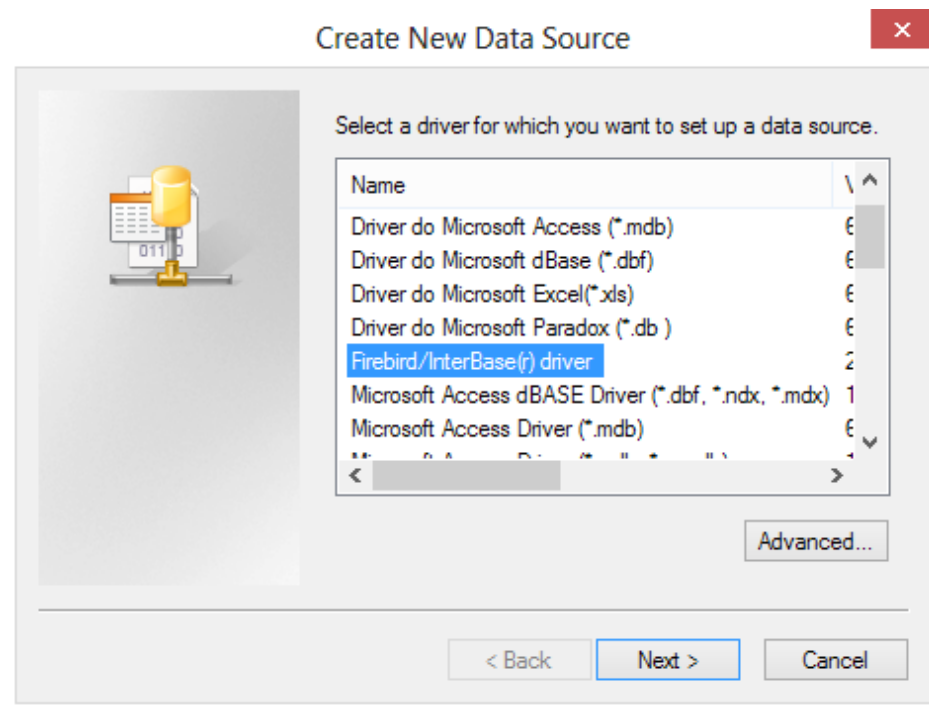
Desnim klikom na željenu bazu podataka (npr., BP-vj10) pojavit će se izbornik u kojemu se nalaze brojne kontrole, između kojih je i *Run a query* (Pokreni upit). Klikom na opciju *Run a Query* otvara se prozor u kojemu se pišu SQL komande kao na sljedećoj slici.



### 10.4.4 Firebird ODBC driver

Kao i većina ostalih SUBP-a, tako i Firebird posjeduje ODBC upravljačke programe za rad s bazom iz vanjskih aplikacija. Uobičajena kolekcija ODBC drivera predinstalirana s Windows operacijskim sustavom dolazi bez podrške za Firebird SUBP te ih je u slučaju potrebe za spajanje aplikacija s Firebird bazom nužno naknadno instalirati s lokacije <http://www.firebirdsql.org/en/odbc-driver/>

Nakon instalacije, upravljački program postaje odmah dostupan među popisom ostalih upravljačkih ODBC programa.



## 10.5 Sadržaj i zadatak vježbe

### 10.5.1 Tijek izvođenja vježbe

1. Preuzmite, instalirajte i konfigurirajte Firebird SUBP te FlameRobin GUI u skladu s uputama iz prethodnih poglavlja.
2. Kreirajte bazu podataka „Raspored“ u Firebird SUBP, tj. preko FlameRobin GUI, korištenjem SQL jezika. Kreirajte minimalno 4 tablice (npr., Dvorana, Predavac, Kolegij, Raspored). Razmislite koje attribute morate definirati za svaku tablicu te kako će one biti međusobno povezane. Također, vodite računa o odgovarajućim tipovima podataka za pojedine attribute.
3. Razmotrite raspored sati za jedan semestar bilo kojeg studijskog smjera Visoke škole te zatečeno stanje prenesite u upravo kreiranu bazu podataka. Pomoću SQL komandi unesite sve potrebne entorke u tablice. Primjerice, za tablicu Dvorana potrebno je unijeti svaku dvoranu u Visokoj školi (1-18, 1-05, 1-06, 2-11, 2-12, 2-18, 2-21, 2-19, 2-22, 1-15).
4. Kreirajte upit kojim ćete prikazati nazive svih dvorana koje su zauzete ponedjeljkom od 10 do 12 sati.
5. Kreirajte upit kojim ćete prikazati nazive dvorana koje su slobodne u utorak od 17 do 19 sati.
6. Kreirajte upit kojim ćete provjeriti koje su dvorane slobodne srijedom od 9 do 11 sati, a u koje stane 60 studenata.

## 10.6 Završna pitanja i zadaci

Nakon aktivnog sudjelovanja i izvođenja vježbe u laboratoriju, potrebno je provjeriti je li vježba savladana u potpunosti. To ćete znati tako da usporedite rezultate izvršenja upita iz zadataka 2 do 5, sa stvarnim stanjem rasporeda sati. Samoprovjerom ćete najlakše detektirati moguće greške u dobivenim rezultatima. Pored toga, poželjno je da samostalno pokušate produbiti i proširiti svoje znanje proučavajući navedene dodatne izvore. Također, pokušajte sami pronaći druge referentne izvore i materijale.

## 10.7 Literatura i dodatni izvori

- <http://www.firebirdsql.org/>
- Firebird 2.5 Quick Start Guide  
[http://www.firebirdsql.org/file/documentation/reference\\_manuals/user\\_manuals/html/qsg25.html](http://www.firebirdsql.org/file/documentation/reference_manuals/user_manuals/html/qsg25.html)
- Firebird 2.5 Language Reference Update  
[http://www.firebirdsql.org/file/documentation/reference\\_manuals/reference\\_material/html/langrefup25.html](http://www.firebirdsql.org/file/documentation/reference_manuals/reference_material/html/langrefup25.html)
- <http://www.vsmi.hr/hr/nastava/raspored-nastave.html>

## Vježba 11: SQL operatori i NULL vrijednost

### 11.1 Motivacija

Razumijevanje SQL-a podrazumijeva i razumijevanje SQL operatora. U SQL-u postoji cijeli niz operatora koji bitno utječu na funkcioniranje komandi u kojima se nalaze. Studente se također upoznaje i s važnim pojmom NULL vrijednosti.

#### Sadržaj vježbe:

<b>VJEŽBA 11:</b>	<b>SQL OPERATORI I NULL VRIJEDNOST</b>	<b>137</b>
11.1	MOTIVACIJA	137
11.2	CILJEVI VJEŽBE – ISHODI UČENJA	138
11.3	SAMOSTALNA PRIPREMA VJEŽBE	138
11.4	TEORIJSKI DIO PRIPREME VJEŽBE	139
11.4.1	SQL operatori.....	139
11.4.2	NULL vrijednost .....	141
11.5	SADRŽAJ ITJEK IZVOĐENJA VJEŽBE	143
11.5.1	Sadržaj vježbe.....	143
11.5.2	Tijek izvođenja vježbe.....	143
11.6	ZAVRŠNA PITANJA I ZADACI	145
11.6.1	Završna pitanja u vezi izložene materije vježbe.....	145
11.6.2	Zadaci za samostalno produbljivanje znanja.....	145
11.7	LITERATURA I DODATNI MATERIJALI	146

## 11.2 Ciljevi vježbe – ishodi učenja

Od studenata se očekuje da, nakon uspješno završene vježbe:

- Shvate pojam i namjenu SQL aritmetičkih, logičkih i poredbenih operatora.
- Razumiju pojam NULL vrijednosti.
- Nauče primjenjivati operatore u do sada naučenim SQL komandama.
- Razumiju posljedice primjene operatora na raznim primjerima.
- Usvoje pojam NULL vrijednosti i posljedice njihove praktične primjene.

## 11.3 Samostalna priprema vježbe

Prije dolaska u lab, obaveze studenata su:

- Proučite prethodnu materiju sa predavanja iz kolegija BP
- Proučite pripremne materijale za ovu vježbu
- Utvrdite prethodnu vježbu – pitanja u prethodnoj vježbi će predstavljati temelj za blic-test u ovoj vježbi
- Minimalna znanja potrebna za pristupanje vježbi su:
  - Znati nabrojati osnovne vrste SQL operatora
  - Razumjeti razlike između SQL operatora
  - Što je NULL vrijednost

## 11.4 Teorijski dio pripreme vježbe

### 11.4.1 SQL operatori

Operatori se definiraju kao rezervirana riječ ili znak a u SQL-u se prvenstveno koriste u sklopu klauzule WHERE radi provođenja aritmetičko-logičkih operacija. Operatori se u SQL-u dijele na sljedeće osnovne skupine:

- Aritmetički operatori
- Operatori usporedbe
- Logički operatori

Aritmetički operatori svrstavaju se u skupinu binarnih operatora jer za njihovu implementaciju potrebno imati dvije suprotstavljene vrijednosti. Postoje, međutim, i aritmetički operatori koji mogu djelovati samo na jedan operand te se svrstavaju i u unarne operatore, poput operacije promjene predznaka.

Aritmetički operatori koji se najčešće koriste u Access-u prikazani su sljedećom tablicom:

Operator	Opis	Primjer
+	zbroj	a+b
-	razlika	Cijena-Popust
*	umnožak	Količina*Cijena
/	dijeljenje	a/b
\	zaokružuje operande na cijele brojeve, te ih dijeli bez ostatka	a\b
Mod	modulo (cjelobrojni ostatak)	a Mod b
^	eksponent	a^b

Primjeri:

```
SELECT 100+200 AS Zbroj;
SELECT 2*3 AS Umnožak;
SELECT Cijena-Popust AS Ukupno;
SELECT Cijena Mod 3=0 FROM Artikl;
```

Operatori usporedbe koriste se pri usporedbi vrijednosti, a kao rezultat vraćaju istinu (TRUE), laž (FALSE) ili NULL vrijednost.

Operatori usporedbe koji se najčešće koriste u Access-u prikazani su sljedećom tablicom:

Operator	Opis	Primjer
<	vraća istinu ako je vrijednost s lijeve strane operatora manja od vrijednosti s desne	vrijednost1 < vrijednost2
<=	vraća istinu ako je vrijednost s lijeve strane operatora	vrijednost1 <= vrijednost2

	manja ili jednaka vrijednosti s desne	
>	vraća istinu ako je vrijednost s lijeve strane operatora veća od vrijednosti s desne	vrijednost1 > vrijednost2
>=	vraća istinu ako je vrijednost s lijeve strane operatora veća ili jednaka vrijednosti s desne	vrijednost1 >= vrijednost2
=	vraća istinu ako je vrijednost s lijeve strane operatora jednaka vrijednosti s desne	vrijednost1 = vrijednost2
<>	vraća istinu ako je vrijednost s lijeve strane operatora različita od vrijednosti s desne	vrijednost1 <> vrijednost2

U slučaju usporedbe s NULL vrijednosti, rezultat usporedbe uvijek će biti NULL bez obzira na vrstu operatora.

Primjeri:

```
SELECT Naziv, Cijena as Djeljivo_s_Tri from Articl where Cijena Mod 3=0;
SELECT * FROM Klijenti WHERE Cijena > 50;
SELECT * FROM Klijenti WHERE Cijena <> 32;
```

Logički operatori koriste se za provjeru istinitosti dvaju izraza, a kao vrijednost vraćaju istinu (TRUE), laž (FALSE) ili NULL vrijednost u slučaju kad je barem jedan od operanada NULL vrijednost.

Logički operatori koji se najčešće koriste u Access-u prikazani su sljedećim tablicama:

Operator	Opis	Primjer
And	vraća logičku vrijednost istina (TRUE) kad su obje vrijednosti istinite	izraz1 And izraz2
Or	vraća logičku vrijednost istina (TRUE) kad je jedna od vrijednosti istina	izraz1 Or izraz2
Eqv	vraća vrijednost istina (TRUE) kad su obje vrijednosti istinite ili kad su obje lažne	izraz1 Eqv izraz2
Not	vraća logičku vrijednost istina (TRUE) kad je izraz na kojeg se primjenjuje laž (FALSE)	izraz1 Not izraz2
Xor	vraća logičku vrijednost istina (TRUE) samo kada je jedan od izraza istina, ali ne oba	izraz1 Xor izraz2

IzrazA	IzrazB	And	Or	Eqv	Xor	Not (IzrazA)	Not (IzrazB)
1	1	1	1	1	0	0	0
1	0	0	1	0	1	0	1
0	1	0	0	0	1	1	0
0	0	0	0	1	0	1	1

IzrazA	IzrazB	And	Or	Eqv	Xor	Not (IzrazA)	Not (IzrazB)
0	NULL	0	NULL	NULL	NULL	1	NULL
1	NULL	NULL	1	NULL	NULL	0	NULL
NULL	0	0	NULL	NULL	NULL	NULL	1
NULL	1	NULL	1	NULL	NULL	NULL	0



NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
------	------	------	------	------	------	------	------

U Access-u se još koriste i operatori specifično namijenjeni tekstualnim vrijednostima, a još se nazivaju i operatori konkatencije.

operator	opis	primjer
&	kombinira vrijednosti dvaju stringova u jedan	'Ivica' & 'Marica'
+	kombinira vrijednost dvaju stringova u jedan uz propagaciju NULL vrijednosti (ako je jedan izraz NULL, tada će cijeli izraz poprimiti NULL)	'Bonnie' + 'Clyde'

U posebnu skupinu operatora ubrajaju se **Is Null**, **Like**, **Between** te **In**, a kao rezultat djelovanja vraćaju vrijednost istine (TRUE) ili laži (FALSE).

operator	opis	Primjer
Is Null (Is Not Null)	provjerava je li izraz NULL	polje1 Is Not Null
Like 'uzorak teksta'	uspoređuje dijelove stringa uz pomoć wildcard znakova ? i *	Ime Like 'M*'
Between vrijednost1 And vrijednost2	provjerava nalaze li se vrijednosti brojeva ili datuma između zadanih rubnih vrijednosti	Cijena Between 50 And 100
In (vrijednost1, vrijednost2, ...)	provjerava nalazi li se neka vrijednost u skupu zadanih vrijednosti	Cijena In (30, 40, 50) Ime In ('Ivo', 'Marta', 'Biba')

Primjeri:

```
SELECT * FROM Klijeti WHERE Dob >= 18 AND Plaća >= 5100;
SELECT * FROM Klijeti WHERE Dob >= 18 OR Plaća >= 5100;
SELECT * FROM Klijeti WHERE Ime LIKE 'Ma*';
SELECT * FROM Klijeti WHERE Dob IN (18, 22, 23);
```

### 11.4.2 NULL vrijednost

Za razliku od leksičkog značenja riječi null proizašle iz engleskog jezika, a koja znači nula ili ništa, u MS Access-u riječ NULL predstavlja nepostojeću ili neprimjenjivu vrijednosti, tzv. No Value. Iz tog se razloga NULL vrijednost ne može koristiti u usporedbama. Također, NULL se razlikuje od stringa duljine nula, iako se prikazom uopće ne razlikuju.

Zbog svoje specifičnosti, u Accessu postoje funkcije koje rade s NULL vrijednostima, a to su: IS NULL, IS NOT NULL i NZ. Funkcija NZ vraća neku zadanu vrijednost u slučaju da je vrijednost varijable koja se provjerava NULL vrijednost.

Sintaksa funkcije NZ je sljedeća:

**Nz (variant, [value\_if\_null])**

Riječ	Opis
<i>variant</i>	vrijednost proizvoljnog podatkovnog tipa
<i>[value_if_null]</i>	vrijednost koja zamjenjuje NULL vrijednost. Ovaj argument je moguće izostaviti a tada se umjesto NULL vrijednosti postavlja nula ili prazan string.

Primjeri:

```
SELECT * FROM Klijenti WHERE Dob IS NOT NULL;  
SELECT [Prezime], [Ime], [Dob], NZ([Adresa], "Adresa nije dostupna")
```

## 11.5 Sadržaj i tijek izvođenja vježbe

### 11.5.1 Sadržaj vježbe

U ovoj vježbi je potrebno otvoriti priloženu bazu podataka **Vj-10-NWT.accdb**

### 11.5.2 Tijek izvođenja vježbe

1. Potrebno je iz tablice Orders (Narudžbe) prikazati ukupne dnevne troškove dostave za sve narudžbe u siječnju 2006. godine.
2. Potrebno je prikazati listu kupljenih proizvoda u tvrtki za naručene količine manje od 40. (referirati se na tablicu Inventory Transactions)
3. Prikazati popis posljednjih inventurnih transakcija s više od 100 jedinica proizvoda. Potrebno je prikazati detalje transakcija za sva dobra koja nisu prodana ('Sold').
4. Iz tablice narudžbe (Orders) prikazati samo one transakcije koje su obavljene u New Yorku putem čekova te u Illinoisu obavljene svim platnim sredstvima osim kreditnim karticama ('Credit card').
5. Prikazati sve transakcije iz Los Angelesa osim onih plaćenih gotovinom te sve transakcije plaćene gotovinom osim u Los Angelesu.
6. Potrebno je provjeriti jesu li sve narudžbe zaključene tj., da su sva plaćanja izvršena gotovinom, čekovima ili kreditnim karticama.
7. Prikazati listu ljudi uključenih u odjel opskrbe pri čemu je u prvom stupcu potrebno prikazati ime, u drugom Prezime a u trećem, odnosno četvrtom sljedeće dva rješenja obzirom na sljedeće zahtjeve:
  - a. NULL vrijednost prikazati kao prazninu, a ostale vrijednosti prikazati kakve jesu.
  - b. Ako je neki od parametara (ime ili prezime) ili oba imaju NULL vrijednost, prikazati ImePrezime također bez vrijednosti
8. Potrebno je prikazati ukupne troškove dostave ([Shipping Fee] + [Taxes]) uz sljedeći uvjet: NULL vrijednosti poreza (Taxes) potrebno je zamijeniti praznim stringom te izračunati ukupni trošak.
9. Provjerite sadržaj tablice OrdersNULL. Uočite kako vrijednosti atributa nekih entorki nisu definirane. Potrebno je napisati upit kojim ćete izdvojiti sve entorke bez definirane vrijednosti atributa [Shipped Date].

10. Kreirajte upit kojim ćete prikazati ukupan promet ( $\text{Quantity} * [\text{Unit Price}]$ ) za proizvode iz kategorije juha (Soups). NAPOMENA: Kombinirajte zapise iz tablica Orders, Order Details i Products.
11. Potrebno je prikazati ukupan promet svih čokoladnih proizvoda (svi u svom nazivu sadrži segment teksta 'chocolate'). NAPOMENA: Kombinirajte zapise iz tablica Orders, Order Details i Products.
12. Potrebno je prikazati ukupan prodaju svih proizvoda koji u svojem nazivu sadrže sljedeći segment teksta: na prvom mjestu je slovo C, drugo slovo je proizvoljno (operator ?), treće slovo je a i četvrto slovo je proizvoljno (operator ?). NAPOMENA: Kombinirajte zapise iz tablica Orders, Order Details i Products.
13. Potrebno je prikazati datum narudžbe (Order Date), naziv proizvoda (Product Name), kategoriju proizvoda (Category) te ukupnu prodaju ( $\text{Quantity} * [\text{Unit Price}]$  as UkupnaProdaja) za sve proizvode iz sljedećih kategorija: Candy, Beverages, Soups, Pasta, Grain. NAPOMENA: Koristite operator IN.

## 11.6 Završna pitanja i zadaci

### 11.6.1 Završna pitanja

Po završetku vježbe, studenti bi trebali bi znati ispravno odgovoriti na pitanja (test će se temeljiti na ovim pitanjima):

- Što su to operatori?
- Koja je uloga operatora u SQL-u?
- Nabrojite operatore u SQL-u.
- Nabrojite aritmetičke operatore i objasnite njihove funkcije.
- Nabrojite logičke operatore i objasnite njihove funkcije.
- Nabrojite poredbene operatore i objasnite njihove funkcije.
- Što je konkatenacija stringova?
- Koji su operatori namijenjeni konkatenaciji stringova?
- Opišite funkciju operatora Is Null.
- Opišite funkciju operatora Is Not Null.
- Opišite funkciju operatora Like.
- Opišite funkciju operatora Between.
- Opišite funkciju operatora In.
- Objasnite značenje NULL vrijednosti.

### 11.6.2 Zadaci za samostalno produblivanje znanja

- Nakon vježbe raspravite s kolegama oko rješenja vježbe na temelju argumenata temeljenih na uvod u vježbu te na temelju predavanja iz kolegija baze podataka.
- Na temelju iskustva stečenog pri rješavanju zadataka ove vježbe pokušajte smisliti još sličnih primjera.

## 11.7 Literatura i dodatni materijali

1. Table of operators (<https://support.office.com/en-us/article/Table-of-operators-e1bc04d5-8b76-429f-a252-e9223117d6bd>)
2. IsNull function (<http://www.techonthenet.com/access/functions/advanced/isnull.php>)
3. Nz Function (<https://support.office.com/en-nz/article/Nz-Function-8ef85549-cc9c-438b-860a-7fd9f4c69b6c>)
4. Like operator ([https://msdn.microsoft.com/en-us/library/bb208897\(v=office.12\).aspx](https://msdn.microsoft.com/en-us/library/bb208897(v=office.12).aspx))

## Vježba 12: SQL agregacije, sortiranje

### 12.1 Motivacija

Ovom vježbom želimo uvesti studente u dio SQL komande koje se tiču funkcija obrade podataka na temelju nekih zajedničkih (grupnih) svojstava primjenom raznih agregatnih funkcija i kriterija koje se postavljaju na dohvaćene skupove. Grupiranje je važan dio SQL operacija u kome se koriste agregatne funkcije. Iako uređenost nije svojstvo relacijskog modela podataka, u SQL-u postoje klauzule kojima je moguće entorke dobivene kao rezultat SQL operacija sortirati prema vrijednostima odabranih atributa.

U ovoj ćemo vježbi na priloženoj bazi podataka te nizu primjera, uvježbavati primjenu agregatnih operacija.

### Sadržaj:

<b>VJEŽBA 12:</b>	<b>SQL AGREGACIJE, SORTIRANJE</b>	<b>  147</b>
12.1	MOTIVACIJA	147
12.2	CILJEVI VJEŽBE – ISHODI UČENJA	148
12.3	SAMOSTALNA PRIPREMA VJEŽBE	148
12.4	TEORIJSKI DIO PRIPREME VJEŽBE	149
12.4.1	<i>Agregatne funkcije u SQL-u</i>	149
12.4.2	<i>Klauzula GROUP BY</i>	151
12.4.3	<i>Klauzula HAVING</i>	152
12.4.4	<i>Klauzula ORDER BY</i>	153
12.5	SADRŽAJ I ZADATAK VJEŽBE	154
12.5.1	<i>Tijek izvođenja vježbe</i>	156
12.6	ZAVRŠNA PITANJA I ZADACI	157
12.6.1	<i>Završna pitanja u vezi izložene materije vježbe</i>	157
12.7	LITERATURA I DODATNI IZVORI	158

## 12.2 Ciljevi vježbe – ishodi učenja

Od studenata se očekuje da nakon uspješno završene vježbe:

- Shvate pojam i namjenu agregatnih funkcija u SQL-u
- Nauče primjenjivati agregatne funkcije u SQL-u
- Shvate pojam i klauzula GROUP BY, HAVING i SORT BY
- Nauče primjenjivati klauzule GROUP BY, HAVING I SORT BY u sklopu komande SELECT

## 12.3 Samostalna priprema vježbe

Obaveza studenata je pripremiti se za za aktivno sudjelovanje u izvođenju vježbe:

- Proučite prethodnu materiju izloženu na predavanjima iz kolegija BP.
- Dobro proučite teorijski dio ove vježbe.
- Potražite dodatne informacije na internetu kojima možete produbiti potrebno znanje.
- Osim teorijske podloge za pripremu ove vježbe, pokušajte shvatiti i ostali dio materijala.



## 12.4 Teorijski dio pripreme vježbe

### 12.4.1 Agregatne funkcije u SQL-u

Izrazi u SELECT komandi mogu između ostalih sadržavati i agregatne funkcije, koje se ponekad nazivaju i statističke ili grupne funkcije. Uz pretpostavku da se u komandi SELECT ne koristi klauzula GROUP BY, agregatna funkcija će tada djelovati na sve atribute što će rezultirati jedim izvedenim atributom čiji podaci ovise o primijenjenoj funkciji. Kaže se da se vrijednosti atributa skupine entorki agregiraju (skupljaju) u jedinstvenu vrijednost te se kao takve i prikazuju. Neke od poznatijih agregatnih funkcija su COUNT, MIN, MAX, SUM, AVG, a sintaksa im je sljedeća:

```
<agregatna funkcija> ::=
  COUNT      ( [ DISTINCT | ALL ] { * | <izraz>
} ) |
  MIN        ( [ DISTINCT | ALL ] < izraz> )
  |
  MAX        ( [ DISTINCT | ALL ] < izraz> )
  |
  SUM        ( [ DISTINCT | ALL ] < izraz> )
  |
  AVG        ( [ DISTINCT | ALL ] < izraz> )
  |
  STDDEV     ( [ DISTINCT | ALL ] < izraz> )
  |
  VARIANCE   ( [ DISTINCT | ALL ] < izraz> )
```

Primjer:

```
select count(*)
from radnik;
```

U ovom primjeru agregatna funkcija COUNT(\*) broji entorke u tablici *radnik*.

U primjeni agregatnih funkcija bez klauzule GROUP BY vrijede sljedeća pravila:

- Upiti s agregatnim funkcijama u svojim izrazima nakon klauzule SELECT rezultiraju samo jednom entorkom
- Nije dozvoljeno ugnježđivanje agregatnih funkcija.
- U upitima s više agregatnih funkcija specifikacija atributa u klauzuli SELECT zahtijeva da se svi atributi navode kao argumenti agregatnih funkcija. Razlog tome nalazi se u prvom pravilu iz kojega slijedi da agregatne funkcije uvijek daju rezultat u obliku jedne vrijednosti dok se klasična specifikacija atributa odnosi na skup vrijednosti.

Agregatne funkcije mogu se koristiti u složenim upitima kao dio svakog podupita.

### 12.4.1.1 Funkcija COUNT

Agregatna funkcija COUNT kao argument može koristiti asterisk (\*) ili proizvoljni izraz. Općenita joj je svrha vratiti broj entorki koje zadovoljavaju zadane kriterije.

Primjeri:

```
select count(SifraLige)
from Igrac;
```

```
select count(all SifraLige)
from Igrac;
```

```
select count(distinct Mjesto)
from Igrac;
```

```
select count(distinct year(datum uplate))
from transakcije;
```

```
select count(distinct(Mjesto), count(distinct(Spol))
from Igrac;
```

### 12.4.1.2 Agregatne funkcije MAX i MIN

Ovim funkcijama moguće je prikazati najveće, odnosno najmanje vrijednosti atributa. Ukoliko neki atribut na kojega se ove funkcije primjenjuju sadrži NULL vrijednost, rezultirajuća vrijednost će također biti NULL vrijednost.

Primjer:

```
select max(Iznos)
from Kazna;
```

```
select min(Iznos)
from Kazna
where SifraIgraca in (select SifraIgraca from Igrac where
Mjesto='Vinkovci');
```

U ovome se primjeru prikazuju najmanje novčane kazne za igrače iz Vinkovaca.

```
select count(*)
from Kazna
where Iznos = (select min(Iznos) from Kazna);
```

U ovome se primjeru prikazuje ukupan broj novčanih kazni jednak onoj najmanjoj.

### 12.4.1.3 Agregatne funkcije SUM i AVG

Funkcija SUM računa sumu vrijednosti nekog atributa dok funkcija AVG vraća srednju vrijednost svih vrijednost nekog atributa. Obje funkcije primjenjive su samo na attribute numeričkog tipa. Na ove

funkcije primjenjuju se ista pravila kao i za prethodne dvije funkcije. Ukoliko su u vrijednostima nekog atributa sadržane samo NULL vrijednosti obje funkcije će vratiti NULL vrijednost. NULL vrijednosti se ne uzimaju u obzir pri sumiranju ili izračunu prosjeka vrijednosti nekog atributa.

Primjeri:

```
select avg(Iznos)
from Kazna
where SifraIgraca=30;
```

```
select SifraKazne, Iznos, abs(Iznos - (select avg(Iznos) from Kazna))
as Razlika
from Kazna;
```

## 12.4.2 Klauzula GROUP BY

Klauzulom GROUP BY grupiraju se entorke na temelju vrijednosti nekih zajedničkih svojstava. Ovako selektirani blokovi zajedničkih entorki mogu biti obrađeni nekom od prethodno opisanih agregatnih funkcija. Tako je npr., moguće saznati koliko je kupljenih proizvoda po određenoj grupi proizvoda ili koliko je prolaznih ocjena po predmetnom nastavniku.

Osnovna sintaksa klauzule GROUP BY je sljedeća:

<klauzula GROUP BY> ::= GROUP BY <specifikacijski niz>

<specifikacijski niz > ::= <specifikacija> [ { ,  
<specifikacija >... ]

< specifikacija> ::= < izraz>

< izraz> ::= <skalarni izraz>

Klauzulu GRUOP BY moguće je primijeniti na grupiranje jednog ili više atributa.

Primjeri:

```
select Mjesto
from Igrac
group by Mjesto;
```

```
select Mjesto, count(*)
from Igrac
group by Mjesto;
```

U ovom primjeru agregatna funkcija count(\*) prebrojava sve igrače grupirane prema mjestu.

```
select SifraTima, ŠifraIgraca
from Utakmica
group by SifraTima, SifraIgraca;
```

Osim grupiranja prema jednom ili više atributa entorke je moguće grupirati i prema proizvoljnim izrazima.

Primjer:

```
select year(datum_placanja), count(*)
from racun
group by year(datum_placanja);
```

U ovome se primjeru entorke grupiraju prema godini izdanog računa (skalarna funkcija year()) te se prebrojavaju prema pojedinoj godini i prikazuju u vidu ukupno plaćenih računa po pojedinoj godini.

U slučaju primjene klauzule GROUP BY na attribute u čijem se skupu vrijednosti pojavljuju i NULL vrijednosti, tada će se takve entorke smatrati zajedničkom grupom.

Prilikom kreiranja upita koji se služe klauzulom GROUP BY valja se voditi sljedećim pravilima:

- Ukoliko komanda SELECT sadrži klauzulu GROUP BY, svaki atribut naveden nakon klauzule SELECT obavezno se mora činiti argument agregatne funkcije ili se nalaziti, a popisu atributa koji se očekuju nakon klauzule GROUP BY.
- Izraze koji se pojavljuju kao parametar pri formiranju grupe entorki, okupljenih oko zajedničke vrijednosti tog parametra, nije nužno navoditi nakon klauzule SELECT jer će se oni kao takvi već pojaviti u prikazu s obzirom na prethodni uvjet (izraz je obavezno navesti nakon klauzule GROUP BY).
- Izrazi koji se pojavljuju kao parametar pri formiranju grupe entorki mogu se koristiti u klauzuli SELECT kao dio složenijih izraza.

Ponavljanje izraza koji se pojavljuju kao parametar pri formiranju grupe entorki nakon klauzule GROUP BY svodi se na minimalni oblik što znači da se ponovljeni izrazi već iskorištenog izraza smatraju nepostojećima. Primjer: GROUP BY smjer, smjer navodi atribut smjer dva puta te se svodi na GROUP BY smjer.

### 12.4.3 Klauzula HAVING

Funkcija klauzule HAVING unutar komande SELECT slična je klauzuli WHERE. Za razliku od klauzule WHERE kojom se prema nekom kriteriju selektiraju entorke nakon klauzule FROM, klauzulom HAVING selektiraju se entorke nakon izvođenja operacije klauzule GROUP BY. Dozvoljena uporaba agregatnih funkcija u klauzuli HAVING bitna je razlika u odnosu na klauzulu WHERE, osim u slučaju primjene podupita.. To znači da se klauzula HAVING može koristiti jedino u kombinaciji s klauzulom GROUP BY i to prema sljedećoj sintaksi:

```
<klauzula having> ::= HAVING <uvjet>
```

Klauzulom HAVING moguće je selektirati grupu rezultatnih entorki nakon operacije grupiranja na temelju vrijednosti svojstava grupe.

Primjer:

```
select year(datum_placanja), count(*)
from racun
group by year(datum_placanja)
having max(year(datum_placanja))=1984;
```

### 12.4.4 Klauzula ORDER BY

Neobavezni dio komande SELECT klauzula je SORT BY, a tiče se uređivanja prikaza entorki u rezultatima upita. Tako je moguće rezultatne entorke poredati uzlanim ili silaznim slijedom prema vrijednosti nekog numeričkog, datumskog ili znakovnog podatkovnog tipa, poput prezimena, datuma rođenja i sl. Sortirati je moguće po vrijednostima jednom ili više atributa te po skalarnim izrazima i atributima koji se nužno ne pojavljuju u popisu izraza i atributa nakon klauzule SELECT. Također, u klauzuli ORDER BY se kao argument mogu primiti, tj. koristiti i podupiti. Sintaksa klauzule ORDER BY je sljedeća:

```
<klauzula ORDER BY> ::=
    ORDER BY <sortiranje> [ { , <sortiranje> }... ]
```

```
<sortiranje> ::=
    <skalarni izraz> [ <smjer sortiranja> ] |
    <naziv atributa> [ <smjer sortiranja> ]
<smjer sortiranja> ::= ASC | DESC
```

Primjeri:

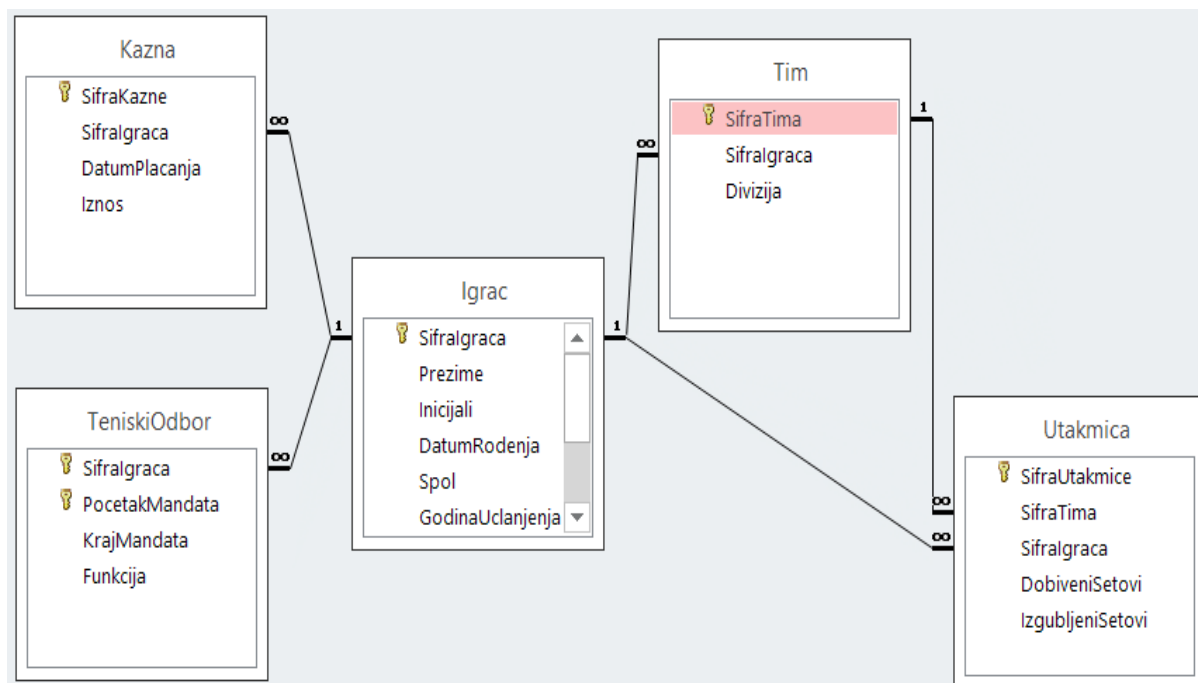
```
select Ime, Prezime
from Radnik
order by Prezime, Dob;
```

```
select SifraIgraca, Iznos
from Kazna as K1
order by (select avg(Iznos) from Kazna as K2
where K.SifraIgraca=K1.SifraIgraca);
```

Sortiranje NULL vrijednosti ovisno je o SQL dijalektu pa se tako ponegdje NULL vrijednost smatra najmanjom vrijednosti u smislu sortiranja, dok u drugim sustavima isto može biti suprotno tumačeno.

## 12.5 Sadržaj i zadatak vježbe

U ovoj vježbi je potrebno otvoriti novu praznu bazu BP VJ 11.accdb čiji je dijagram veza među relacijama zadan sljedećom slikom.



Zadana je baza podataka koja sadrži podatke teniskog kluba osnovanog 1970. godine. Baza podataka se sastoji od 5 tablica: Igrac, Tim, TeniskiOdbor, Utakmica, Kazna. Tablica Igrač sadrži podatke o igračima članovima teniskog kluba. Teniski klub provodi upise jednom godišnje, točnije 1.1., počevši od 1990. godine. Svi igrači koji odustanu od članstva brišu se iz tablice Igrač. Teniski klub podijeljen je na dvije grupe članova – igrača: rekreativce i natjecatelje. Rekreativni članovi mogu igrati mečeve jedino međusobno, a podaci o njihovim susretima ne bilježe se u bazi. Natjecateljski igrači igraju u timovima protiv igrača drugih klubova, a rezultati tih mečeva bilježe se u bazu. Svaki igrač, bez obzira kojoj grupi pripada, ima svoj jedinstveni broj u klubu, takozvanu šifru igrača, dok natjecateljski članovi kluba moraju biti registrirani i u teniskoj ligi koja im dodjeljuje jedinstveni ligaški broj. U slučaju da natjecateljski igrač prestane natjecateljsku karijeru i postane rekreativac, briše mu se jedinstveni ligaški broj. Klub raspolaže određenim brojem timova koji sudjeluju u natjecanjima. Kapetan svakog tima, kao i divizija u kojoj se trenutno natječe bilježe se u bazu podataka. Kapetan ne mora nužno igrati utakmicu za svoj tim te je moguće da jedan igrač bude kapetan više timova istovremeno. U slučaju da je tim premješten u drugu diviziju, zapisi o timu se brišu i pune novim podacima. Isto vrijedi za kapetane timova. Zamjena kapetana podrazumijeva promjenu brojeva kapetanskih igrača. Tim se sastoji određenog broja igrača. Kada tim igra protiv tima iz drugog teniskog kluba, svaki igrač u timu igra protiv igrača suprotstavljenog tima (dakle, nema miješanja već se igra „naši protiv vaših“). Tim se smatra pobjedničkim ako ima najviše pojedinačnih pobjeda svojih igrača. Tim se ne mora uvijek sastojati od istih igrača te su moguće zamjene

rezervnim igračima tako što jedan igrač može nastupati za više timova. Meč je podijeljen na igru u setovima, a igrač koji osvoji veći broj setova smatra se pobjedničkim. Uobičajeno, igra završava nakon što jedan od igrača prvi osvoji 2 ili 3 seta, ovisno o prethodnom dogovoru. Krajnji doseg igre u 2 dobivena seta je igra u 21, odnosno 20 setova, dok je za igru u 3 dobivena seta maksimalni doseg 32, 31 ili 30 setova. Svaki meč mora završiti pobjedom ili porazom. U tablici Utakmica bilježe se za svakog igrača podaci o odigranim mečevima. U slučaju neprimjerenog ponašanja igrača što uključuje kašnjenje na mečeve ili otkazivanje mečeva bez najave, agresivno ponašanje na terenu i izvan i sl., izriče mu se kazna u obliku novčane globe, a podaci o tim događajima bilježe se u tablicu Kazna. Zapisi o kaznama pojedinih igrača čuvaju se u bazi sve dok igrač igra u natjecateljskoj ligi. Ako igrač napusti klub brišu se svi njegovi podaci u svih 5 tablica. Ako klub ukine neki tim brišu se podaci tog tima u tablicama Tim i Utakmica. Ako član kluba prestane biti natjecatelj i postane rekreativac brišu se zapisi svih mečeva te kazni za tog igrača. Od 1.1.1990. u tablici TeniskiOdbor bilježe se podaci o članstvu u odboru na jednoj od četiri pozicije: predsjednik, tajnik, blagajnik i član. Novi članovi odabiru se 1. siječnja svake godine.

### 12.5.1 Tijek izvođenja vježbe

1. Koliko je igrača zapisano u tablici Igrač?
2. Koliko igrača živi u Vukovaru?
3. Koliko se ukupno kazni nalazi zapisano u tablici kazne te koji je najviši iznos kazne?
4. Koliko je ukupno brojeva lige dodijeljenih igračima?
5. Koliko se različitih gradova nalazi u tablici Igrač?
6. Koliko je ukupno različitih početnih slova prezimena igrača u tablici Igrač? (Koristite skalarnu funkciju `substr()` ).
7. Koliko je različitih godina u tablici Kazna? (Koristite skalarnu funkciju `year()` ).
8. Prikazati broj različitih funkcija u tablici TeniskiOdbor.
9. Kolika je najviša kazna zabilježena u tablici Kazna?
10. Koja je razlika između iznosa najviše i najniže kazne u lipama?
11. Koji je najveći ligaški broj za igrače iz Osijeka?
12. Prikazati najmanji broj setova kojima je dobiven meč.
13. Prikazati prosječan iznos kazne za igrača sa šifrom 44.
14. Prikazati popis različitih naziva gradova iz tablice Igrač.
15. Prikazati naziv grada te ukupan broj igrača u tom gradu.
16. Prikazati broj tima, broj odigranih utakmica te ukupan broj dobivenih setova za svaki tim.
17. Prikazati različite iznose kazni, broj ponavljanja istih te ukupan iznos iste kazne (iznos kazne \* broj ponavljanja).
18. Prikazati popis različitih godina u kojima su se igrači pridruživali klubu.
19. Prikazati ukupan broj igrača prema godini u kojoj su se pridružili klubu.
20. Prikazati šifru igrača, prosječan iznos kazne te ukupan broj kazni za sve igrače koji su zaradili barem jednu kaznu.
21. Prikazati sve kombinacije brojeva tima te brojeva igrača iz tablice Utakmica.
22. Prepravite upit iz prošlog zadatka te prikažite i ukupan broj dobivenih setova, broj utakmica te najmanji broj izgubljenih setova.
23. Prikažite šifru igrača, prezime te ukupan iznos zarađenih kazni za sve igrače koji su zaradili barem jednu kaznu.
24. Prikazati broj dobivenih utakmica za svaku kombinaciju dobivenih i izgubljenih setova.
25. Prikazati ukupan broj dobivenih setova s obzirom na grad i diviziju.
26. Prikazati prezime i broj zarađenih kazni za igrače iz Križevaca.
27. Prikazati šifru tima, diviziju i ukupan broj dobivenih setova za svaki tim.
28. Prikazati broj plaćenih kazni za svaku godinu iz tablice Kazna.



29. Prikazati razliku između broja dobivenih i izgubljenih setova koristeći funkciju ABS te prikazati rezultate grupirane prema toj razlici.
30. Prikazati različite brojeve lige dodijeljene igračima.
31. Prikazati šifre igrača koji su zaradili više od jedne kazne.
32. Prikazati šifre igrača čije su posljednje kazne zarađene 1984. godine.
33. Prikazati šifru igrača i ukupan iznos kazni za sve one koji su prikupili kazni ukupnog iznosa većeg od 150 kn.
34. Prikazati nazive gradova u kojima živi više od 4 igrača.
35. Prikazati sve iznose kazni zabilježenih u tablici Kazna te ih poredajte prema šifri igrača uzlazno te iznosu kazne silazno.

## 12.6 Završna pitanja i zadaci

Nakon aktivnog sudjelovanja i izvođenja vježbe u labu, potrebno je provjeriti je li vježba shvaćena u potpunosti. To ćete znati ako ste po završetku vježbe, u stanju samostalno i ispravno odgovoriti na završna pitanja (test u idućoj vježbi će se temeljiti na ovim pitanjima). Pored toga, poželjno je da samostalno pokušate produbiti i proširiti svoje znanje proučavajući navedene dodatne izvore. Također, pokušajte sami pronaći druge referentne izvore i materijale.

### 12.6.1 Završna pitanja u vezi izložene materije vježbe

- Što su to agregatne funkcije?
- Nabrojite najčešće agregatne funkcije korištene u SQL-u.
- Opišite ulogu agregatne funkcije COUNT.
- Opišite ulogu agregatne funkcije MAX.
- Opišite ulogu agregatne funkcije MIN.
- Opišite ulogu agregatne funkcije AVG.
- Koja je uloga klauzule GROUP BY u SELECT komandi?
- Koja je uloga klauzule HAVING u SELECT komandi?
- Usporedite klauzule WHERE i HAVING. Koje su razlike?
- Koja je uloga klauzule ORDER BY u SELECT komandi?

## 12.7 Literatura i dodatni izvori

1. [ORDER BY clause \(https://support.office.com/en-za/article/ORDER-BY-Clause-e8ea47f7-5388-460a-bec8-dcc81792d762\)](https://support.office.com/en-za/article/ORDER-BY-Clause-e8ea47f7-5388-460a-bec8-dcc81792d762)
2. [GROUP BY clause \(https://support.office.com/en-GB/article/GROUP-BY-Clause-84eeb766-25d2-4aa1-8eea-002bb65ef3a0\)](https://support.office.com/en-GB/article/GROUP-BY-Clause-84eeb766-25d2-4aa1-8eea-002bb65ef3a0)
3. [HAVING clause \(https://support.office.com/en-GB/article/HAVING-Clause-64c52dba-5cda-45c5-98b5-bd155a89f02f\)](https://support.office.com/en-GB/article/HAVING-Clause-64c52dba-5cda-45c5-98b5-bd155a89f02f)
4. [SQL Aggregate Functions \(https://msdn.microsoft.com/en-us/library/office/ff197054.aspx\)](https://msdn.microsoft.com/en-us/library/office/ff197054.aspx)
5. [WHERE clause \(https://support.office.com/en-nz/article/Access-SQL-WHERE-clause-753bbc13-debc-4b28-b527-42eb7885c862\)](https://support.office.com/en-nz/article/Access-SQL-WHERE-clause-753bbc13-debc-4b28-b527-42eb7885c862)

## Vježba 13: Ugniježdjeni upiti

### 13.1 Motivacija

Do sada smo u primjerima SQL komandi koristili jednostruke upite. U ovoj vježbi ćemo obraditi pojam podupita i pokazati kako se upiti mogu ugniježditi. SQL podupiti omogućuju izgradnju kompleksnijih SQL komandi.

#### Sadržaj vježbe:

<b>VJEŽBA 13:</b>	<b>UGNIJEŽDjeni UPITI</b>	<b>159</b>
13.1	MOTIVACIJA	159
13.2	CILJEVI VJEŽBE – ISHODI UČENJA	160
13.3	SAMOSTALNA PRIPREMA VJEŽBE	160
13.4	TEORIJSKI DIO PRIPREME VJEŽBE	161
13.4.1	<i>Ugniježdjeni upiti - uvod</i>	161
13.5	SADRŽAJ I TIJEK IZVOĐENJA VJEŽBE	163
13.5.1	<i>Tijek izvođenja vježbe</i>	163
13.6	ZAVRŠNA PITANJA I ZADACI	169
13.6.1	<i>Završna pitanja u vezi izložene materije vježbe</i>	169
13.7	LITERATURA I DODATNI MATERIJALI	170

## 13.2 Ciljevi vježbe – ishodi učenja

Od studenata se očekuje da, nakon uspješno završene vježbe:

- Shvate pojam i namjenu ugniježdenih upita.
- Nauče primjenjivati ugniježdene upite u SQL-u.
- Razumiju posljedice primjene ugniježdenih upita.

## 13.3 Samostalna priprema vježbe

Prije dolaska u lab, obaveze studenata su:

- Proučite prethodnu materiju sa predavanja iz kolegija BP
- Proučite pripremne materijale za ovu vježbu
- Utvrdite prethodnu vježbu – pitanja u prethodnoj vježbi će predstavljati temelj za blic-test u ovoj vježbi
- Minimalna znanja potrebna za pristupanje vježbi su:
  - Poznavati osnovnu sintaksu izraza SELECT
  - Poznavati osnovne SQL operatore i njihovu pravilnu primjenu

## 13.4 Teorijski dio pripreme vježbe

### 13.4.1 Ugniježdeni upiti - uvod

Select komanda ugniježđena unutar klauzule WHERE neke druge, SELECT, UPDATE ili DELETE komande vanjskog upita, naziva se podupit (subquery). Svaki podupit piše se unutar okruglih zagrada. Upit koji uključuje podupit (subquery), svoj uvjetni dio izraza temelji na rezultatima koji su dobiveni izvršenjem podupita. Pri tome podupit i glavni upit mogu djelovati na istu tablicu ili na različite tablice. Mnogi SQL izrazi koji uključuju podupite mogu se realizirati i stvaranjem upita sa nekom vrstom spoja (join).

```
SELECT [ALL | DISTINCT] nazivi atributa, agregatni atributi FROM
nazivi_tablica| join veze [WHERE naziv_atributa predikat
(podupit)] [GROUP BY nazivi atributa] [HAVING naziv_atributa
predikat (podupit)]
```

Sintaksa podupita je sljedeća:

```
SELECT [ALL | DISTINCT] nazivi atributa, agregatni atributi FROM
nazivi_tablica| join veze [WHERE izraz uvjeta] [GROUP BY nazivi
atributa] [HAVING izraz uvjeta]
```

Prilikom izvršenja SQL izraza koji uključuje podupit, najprije se izvršava podupit, a rezultati koje podupit daje postaju dio izraza uvjeta glavnog upita. Primjena podupita podrazumijeva postojanje određenih ograničenja i pravila:

- U listi atributa koji se dohvaćaju kao rezultat podupita može biti sadržan naziv samo jednog atributa, osim u slučajevima primjene podupita postojanja (Exists).
- Podupiti s operatorom usporedbe moraju uvijek davati samo jednu vrijednost te ne mogu uključivati grupiranje (GROUP BY i HAVING klauzulu).
- Modifikator DISTINCT ne može se primijeniti sa podupitima koji uključuju GROUP BY izraz.
- Podupit, tj. select izraz kojim se podupit definira uvijek mora biti naveden u zagradama.
- U sklopu podupita nije moguće koristiti izraz ORDER BY

Postoje tri osnovna oblika podupita (subqueries):

- podupiti liste na koje se primijenjuje IN predikat
- podupiti sa predikatom usporedbe
- podupiti kojima se ispituje postojanje određenih podataka.

### 13.4.1.1 Podupiti liste

Kod ove vrste podupita, niz rezultata vraća se u obliku liste, koju glavni upit koristi u svom uvjetnom izrazu preko predikata liste (IN ili NOT IN).

### 13.4.1.2 Podupiti s predikatom usporedbe

Rezultat podupita može se u glavni upit uključiti preko jednog od predikata usporedbe (=,<>,<,...)

Takva primjena podupita podrazumijeva da podupit mora vratiti samo jednu vrijednost, a ne kao u prethodnom primjeru listu vrijednosti. Ukoliko podupit vraća više od jedne vrijednosti javlja se greška u izvršavanju glavnog upita.

Poseban slučaj podupita sa predikatom usporedbe predstavlja primjena modifikatora ANY - bilo koji i ALL - svi. U slučaju primjene ovih modifikatora podupiti sa predikatom usporedbe mogu vraćati listu vrijednosti, a ne samo jednu vrijednost. Ovi modifikatori primjenjuju se u obliku

```
SELECT [ALL | DISTINCT] nazivi atributa, agregatni atributi FROM
nazivi_tablica| join veze WHERE naziv_atributa
predikat_usporedbe ANY| ALL (podupit)]
```

Na primjer, operator > (veći od) primijenjen s modifikatorom ALL , >ALL znači veći od svih vrijednosti koje se nalaze u listi, tj. veći od maksimalne vrijednosti iz liste. >ALL (1,2,3) znači veći od 3.

Modifikator ANY znači bilo koji (odnosno barem jedan). > ANY znači veći od bilo kojeg, tj. veći od najmanje vrijednosti u listi. > ANY (1,2,3) znači veći od 1.

### 13.4.1.3 Korelirani podupiti

U upitima koji uključuju korelirane podupite, izračunavanje podupita zavisi od glavnog upita, što znači da se podupit izvodi za svaki podatak glavnog upita. Osnovna razlika ovog tipa podupita u odnosu na uobičajene ugniježdene podupite, je u načinu njihova izvršavanja, gdje se ugniježđeni SELECT izrazi referiraju na tablice u prethodnom ili vanjskom SELECT izrazu. Vrijednosti koje korelirani podupiti dohvaćaju zavisne su od vrijednosti varijabli iz vanjskih upita na koje se u svojoj deklaraciji pozivaju. Takvi se upiti neprestano ponavljaju za svaku vrijednost varijable vanjskog upita, čime se bitno razlikuju od uobičajenih ugniježđenih upita gdje se podupiti kao takvi izvršavaju samo jednom. Tako se u

sljedećem primjeru za svaki kandidatnu entorku vanjskog SELECT izraza, koja odgovara uvjetu klauzule WHERE u drugom SELECT izrazu, izvršava podupit kako je navedeno.

Primjer:

```
SELECT jmbag
FROM student AS s
WHERE 1>
(SELECT count(*) FROM ispit AS i
WHERE i.sifra_ispita=a.sifra_ispita);
```

#### 13.4.1.4 Podupiti s predikatom postojanja (EXISTS, NOT EXISTS)

Operator EXISTS ispituje postojanje podataka uz navedene uvjete i najčešći je način korištenja koreliranih podupita. U ovakvoj strukturi ugniježđenog upita, uvjetni izraz glavnog upita ispituje postojanje podataka koje vraća podupit. Uvjetni izraz glavnog upita provjerava postojanje podataka u podupitu. Podupit s predikatom postojanja ne vraća nikakve podatke, već daje samo rezultat TRUE ili FALSE.

## 13.5 Sadržaj i tijek izvođenja vježbe

### 13.5.1 Tijek izvođenja vježbe

U ovoj vježbi je potrebno otvoriti bazu podataka iz prethodne vježbe te na osnovu priloženog teorijskog uvoda riješiti sljedeće zadatke.

1. Kreirati upit kojim ćete prikazati šifru igrača te prezime za one igrače koji su zaradili više kazni od broja odigranih mečeva.

Sifralgraca	Prezime
27	Gašpar
44	Ban

2. Kreirati upit kojim ćete prikazati šifru igrača, prezime te broj zarađenih kazni za one igrače koji imaju najmanje dvije kazne.

Sifralgraca	Prezime	broj
27	Gašpar	2
44	Ban	3

3. Kreirati upit kojim ćete prikazati ukupan broj kazni te ukupan broj utakmica.
4. Kreirati upit kojim ćete prikazati broj različitih pozicija u teniskom odboru.

Expr1000
4

5. Kreirati upit kojim ćete prikazati šifru tima, diviziju i ukupan broj mečeva odigranih za taj tim.

SifraTima	Divizija	Expr1002
1	prva	8
2	druga	5

6. Kreirati upit kojim ćete prikazati šifru igrača, prezime te broj dobivenih mečeva.

Sifralgraca	Prezime	Expr1002
2	Kovačić	0
6	Bošković	2
7	Atkins	0
8	Cvitković	0
27	Gašpar	1
28	Mirković	0
39	Babić	0
44	Ban	1
57	Britvić	1
83	Hrvatini	0
95	Miler	0
100	Primorac	0
104	Murtić	1
112	Bajs	0

7. Kreirati upit kojim ćete dobiti sljedeću tablicu:

STAVKA	uk
Ukupno igrača	14
Ukupno timova	2
Ukupno utakmica	13

8. Kreirati upit kojim ćete prikazati najniži iznos kazne koju je zaradio igrač iz Vukovara.

Expr1000
100,00 kn

9. Kreirati upit kojim ćete prikazati koliko je ukupno kazni iznosa jednakog najmanjem.

Expr1000
2

10. Kreirati upit kojim ćete prikazati šifru tima te šifru igrača koji je dobio najviše mečeva za taj tim.

SifraTima	Sifralgraca
1	6
1	44
1	57
2	27
2	104

11. Kreirati upit kojim ćete prikazati šifru igrača, najviši iznos kazne i datum plaćanja te kazne za one igrače koji su zaradili najmanje jednu kaznu.



Sifralgraca	Iznos	DatumPlacar
6	100,00 kn	8.12.1980.
44	75,00 kn	5.5.1981.
27	100,00 kn	10.9.1983.
104	50,00 kn	8.12.1984.
8	25,00 kn	8.12.1980.

12. Kreirajte upit kojim ćete prikazati šifru igrača, najviši iznos plaćene kazne i najveći broj dobivenih setova u nekom meču.

Sifralgraca	najvisaKazna	BrojSetova
2		1
6	100,00 kn	3
7		
8	25,00 kn	0
27	100,00 kn	3
28		
39		
44	75,00 kn	3
57		3
83		0
95		
100		
104	50,00 kn	3
112		2

13. Kreirati upit kojim ćete prikazati šifre svih igrača kojima su jednaki iznosi najmanje i najviše kazne.

Sifralgraca
6
8
104

14. Kreirati upit kojim ćete prikazati šifru igrača te razliku njegove najniže i najviše kazne.

Sifralgraca	Expr1001
2	
6	0
7	
8	0
27	25
28	
39	
44	50
57	
83	
95	
100	
104	0
112	

15. Kreirati upit kojim ćete prikazati šifru i datum rođenja igrača rođenih iste godine kao i najmlađi igrač koji je igrao za tim broj 1.

Sifralgraca	DatumRoder
57	17.8.1971.

16. Kreirati upit kojim ćete prikazati ukupan iznos kazni za igrače iz Križevaca.

Expr1000
155,00 kn

17. Kreirati upit kojim ćete prikazati sve igrače koji su zaradili kazne u iznosu većem od prosječnog.

Sifralgraca
6
27
44

18. Kreirati upit kojim ćete prikazati šifru kazne, iznos i razliku iznosa i srednje vrijednosti kazni.

SifraKazne	Iznos	Razlika
1	100,00 kn	40
2	75,00 kn	15
3	100,00 kn	40
4	50,00 kn	10
5	25,00 kn	35
6	25,00 kn	35
7	30,00 kn	30
8	75,00 kn	15

19. Kreirati upit kojim ćete prikazati prosječnu kaznu igrača koji su igrali ili još igraju za tim broj 1.

Expr1000
51,00 kn

20. Kreirati upit kojim ćete prikazati šifre i prezimena igrača kojima je ukupan iznos kazni veći od 100kn.

Sifralgraca	Prezime
27	Gašpar
44	Ban

21. Kreirati upit kojim ćete prikazati prezimena igrača koji su u najmanje jednom odigranom meču osvojili više setova nego što ih je ukupno osvojio igrač 27.

Prezime
Bošković
Ban
Britvić
Murtić

22. Kreirati upit kojim ćete prikazati šifre i prezimena igrača kojima je suma svih dobivenih setova jednaka 8.

Sifralgraca	Prezime
6	Bošković

23. Kreirati upit kojim ćete prikazati prezimena igrača kojima je duljina prezimena veća od prosječne duljine prezimena. Koristite sljedeće funkcije: LEN() – funkcija vraća duljinu primljenog stringa; RTRIM() – funkcija uklanja eventualne praznine na kraju primljenog stringa.

Sifralgraca	Prezime
2	Kovačić
6	Bošković
8	Cvitković
28	Mirković
57	Britvić
83	Hrvatinić
100	Primorac

24. Kreirati upit kojim ćete prikazati šifru igrača i razliku maksimalne i prosječne kazne za sve igrače (i one koji nemaju kazne).

Sifralgraca	Expr1001
2	
6	0
7	
8	0
27	12,5
28	
39	
44	31,6667
57	
83	
95	
100	
104	0
112	

25. Kreirati upit kojim ćete prikazati šifru tima i ukupan broj mečeva odigranih za taj tim i za sve timove kojima je kapetan igrač iz Virovitice.

SifraTima	Expr1001
2	5

26. Kreirati upit kojim ćete prikazati šifru tima, ukupan broj mečeva i ukupan broj dobivenih setova za sve timove iz prve lige.

SifraTima	Expr1001	Expr1002
1	8	15

27. Kreirati upit kojim ćete prikazati šifru i ukupan iznos kazni igrača koji su imali ili imaju kapetansku funkciju te su zaradili kazni u ukupnom iznosu većem od 80kn.

Sifralgraca	Expr1001
6	100,00 kn
27	175,00 kn

28. Kreirati upit kojim ćete prikazati šifru i ukupan iznos kazni za igrače s najvišim iznosom pojedinačne kazne.

Sifralgraca	Expr1001
27	175,00 kn

29. Kreirati upit kojim ćete prikazati šifru tima te ukupan broj igrača tog tima za kojega je igrala većina igrača iz tablice Igrač.

SifraTima	Expr1001
1	8

30. Kreirati upit kojim ćete prikazati šifru tima i diviziju za svaki tim u kojemu se natjecalo više od 4 igrača.

Divizija	SifTima
prva	1

31. Kreirati upit kojim ćete prikazati prezimena igrača koji su zaradili 2 ili više kazni većih od 40kn.

Prezime
Gašpar

32. Kreirati upit kojim ćete prikazati prezimena igrača s najvišim iznosom ukupnih kazni.

Prezime
Gašpar

33. Kreirati upit kojim ćete prikazati šifre igrača koji su zaradili dvostruko veće iznose kazni od igrača 104.

Sifralgraca
6

34. Kreirati upit kojim ćete prikazati šifre igrača koji su prikupili jednak broj kazni kao igrač 6.

Sifralgraca
8
104

## 13.6 Završna pitanja i zadaci

### 13.6.1 Završna pitanja

Po završetku vježbe, studenti bi trebali bi znati ispravno odgovoriti na pitanja (test će se temeljiti na ovim pitanjima):

- Što su to podupiti?
- Koja su ograničenja primjene podupita?
- Pod kojim se klauzulama mogu primijeniti podupiti?
- Što su korelirani podupiti?

## 13.7 Literatura i dodatni materijali

1. [SQL Subqueries \(https://msdn.microsoft.com/en-us/library/office/ff192664.aspx\)](https://msdn.microsoft.com/en-us/library/office/ff192664.aspx)
2. [Access Subquery Techniques \(https://msdn.microsoft.com/en-us/library/office/aa217680\(v=office.11\).aspx\)](https://msdn.microsoft.com/en-us/library/office/aa217680(v=office.11).aspx)

## Vježba 14: MySQL DBMS, MySQL Workbench |

### 14.1 Motivacija

MySQL DBMS je jedna od najraširenijih SUBP-a. Njegova upotreba je osobito prisutna kod web aplikacija. MySQL pripada kategoriji softvera otvorenog koda i kao takav je dostupan širokom krugu programera, osobito za web aplikacije. Smatrali smo stoga da je nužno da ga uključimo u ovu vježbu.

Danas je MySQL u vlasništvu korporacije ORACLE, koja je radi njegove popularizacije izdala MySQL Workbench. MySQL Workbench je integrirani alat koji omogućuje vizualni razvoj modela podataka odnosno sheme baze, administraciju, generiranje SQL upita, održavanje i kopiranje bazepodataka itd. Zbog toga je svakom razvijaju-programeru koji koristi MySQL nepohodan i MySQL Workbench.

Ovom vježbom želimo studentima pokazati perspektivu primjene SQL-a na primjeru složenijih SQL upita u okruženju popularnog MySQL SUBP-a.

#### Sadržaj vježbe:

VJEŽBA 14: MySQL SUBP, MYSQL WORKBENCH		171
14.1	MOTIVACIJA	171
14.2	CILJEVI VJEŽBE – ISHODI UČENJA	172
14.3	SAMOSTALNA PRIPREMA VJEŽBE	172
14.4	TEORIJSKI DIO PRIPREME VJEŽBE	173
14.4.1	<i>Uvod u MySQL SUBP.....</i>	<i>173</i>
14.4.2	<i>MySQL instalacija.....</i>	<i>174</i>
14.4.3	<i>MySQL Workbench.....</i>	<i>185</i>
14.4.4	<i>Uvoz podataka iz CSV datoteka.....</i>	<i>195</i>
14.5	SADRŽAJ I TIJEK IZVOĐENJA VJEŽBE	198
14.5.1	<i>Tijek izvođenja vježbe.....</i>	<i>198</i>
14.6	ZAVRŠNA PITANJA I ZADACI	202
14.6.1	<i>Zadaci za samostalno produbljivanje znanja.....</i>	<i>202</i>
14.7	LITERATURA I DODATNI MATERIJALI	202

## 14.2 Ciljevi vježbe – ishodi učenja

Od studenata se očekuje da, nakon uspješno završene vježbe:

- Nauče kako se instalira MySQL DBMS
- Nauče kako se instalira MySQL Workbench
- Nauče kako se pomoću MySQL Workbench-a kreira/mijenja shema baze podataka
- Nauče kako se pomoću MySQL Workbench-a manipulira s podacima u bazi podataka (uvoz, izvoz podataka, itd.)
- Nauče primjenjivati do sada stečene vještine na složenije slučajeve SQL upita
- Upoznaju još jedan sustav za upravljanje relacijskim bazama podataka, odnosno njegove specifičnosti

## 14.3 Samostalna priprema vježbe

Prije dolaska u lab, obaveze studenata su:

- Proučite prethodnu materiju sa predavanja iz kolegija BP
- Proučite pripremne materijale za ovu vježbu
- Utvrdite prethodnu vježbu – pitanja u prethodnoj vježbi će predstavljati temelj za blic-test u ovoj vježbi



## 14.4 Teorijski dio pripreme vježbe


### 14.4.1 Uvod u MySQL SUBP

U ovoj vježbi ukratko će se prezentirati MySQL sustav za upravljanje bazama podataka (SUBP, odnosno DBMS). Radi se o sustavu koji je javnosti predstavljen 1995. godine, tada još u vlasništvu tvrtke MySQL AB te se kao takav razvijao sve do 2010. godine od kada se nalazi u vlasništvu tvrtke Oracle. S obzirom na povijest nastanka i razvoja, do današnjih dana ostao je neizostavni dio softverske platforme AMP (Apache, MySQL, Perl/PHP/Python) namijenjene razvoju web aplikacija te je upravo u toj niši stekao najveću slavu. Tako je usvojen i od strane poznatih web CMS sustava kao što su Joomla!, TYPO3, WordPress, Drupal i dr.

U skladu s brojnim formalnim i neformalnim domaćim i svjetskim usmenim i pismenim izvorima, te isto tako primjenom u brojnim privatnim i poslovnim praktičnim aplikacijama (Facebook, Twitter, Flickr, Youtube i sl.), MySQL se svrstava u sam vrh po popularnosti, a potvrda toga je i drugo mjesto prema posljednjem izvješću db-engines portala za mjerenje popularnosti DBMS sustava. Razlozi su brojni, no mogu se, prema zadanim parametrima za rangiranje popularnosti navedenih sustava svrstati u sljedeće:

- broj upita u Google i Bing tražilicama, oblika „naziv sustava database“
- istraživanje općeg interesa za pojedine sustave putem usluge Google Trend
- učestalost rasprava o pojedinim sustavima na popularnim specijaliziranim forumima
- broju ponuđenih poslova u kojim se kao kriterij spominje poznavanje neki DBMS sustav
- broju profila na poslovnim društvenim mrežama koji se povezuju s DBMS sustavima
- popularnosti na društvenim mrežama

238 systems in ranking, December 2014

Rank	Last Month	DBMS	Database Model	Score	Changes
1.	1.	Oracle	Relational DBMS	1459.79	+7.67
2.	2.	MySQL	Relational DBMS	1268.58	-10.50
3.	3.	Microsoft SQL Server	Relational DBMS	1200.05	-20.15
4.	4.	PostgreSQL	Relational DBMS	254.01	-3.35
5.	5.	MongoDB	Document store	246.52	+1.78
6.	6.	DB2	Relational DBMS	210.25	+4.02
7.	7.	Microsoft Access	Relational DBMS	139.89	+1.06
8.	8.	SQLite	Relational DBMS	94.70	-0.58
9.	9.	Cassandra	Wide column store	94.06	+2.07
10.	 11.	Redis	Key-value store	87.88	+5.53

Isječak rang liste popularnosti DBMS sustava prema portalu db-engines.com (Izvor: <http://db-engines.com/en/ranking>)

Osim što se svrstava uz bok najmoćnijem komercijalnom konkurentu – Oracle DBMS, ujedno je i najpopularniji open source RDBMS, te se kao sustav otvorenog koda od 2000. godine razvija i pruža pod uvjetima dualne licence: besplatnom GNU GPL licencom (MySQL Community Edition) te pod raznim komercijalnim licencama koje odgovaraju specifičnim potrebama tržišta (MySQL Standard Edition, MySQL Enterprise Edition te MySQL Cluster Carrier Grade Edition).

- Na službenim stranicama distributora moguće je doznati da se besplatna verzija MySQL-a, između ostaloga, odlikuje i sljedećim svojstvima:
- višestruki upravljački mehanizmi (engines): InnoDB, MyISAM, NDB, i dr.
- podrška za master-slave asinkronu replikaciju radi povećanja skalabilnosti i pouzdanosti sustava
- podrška za particioniranje radi povećanja performansi brzine i dostupnosti u prilikama višekorisničkog pristupa
- podrška za spremljene procedure radi poboljšanja performansi i olakšanog razvoja aplikacija
- podrška za trigere i pogleda
- nadzor rada MySQL poslužitelja na razini fizičkih performansi (performance schema)
- pristup upravljanju metapodacima u bazi
- podrška za ODBC, JDBC i .NET upravljačke programe za razvoj aplikacija u različitim programskim okruženjima
- vizualne alate za upravljanje sustavom za relacijske baze podataka
- multiplatformska podrška

MySQL DBMS standardno se isporučuje bez vizualnih ili komandnih alata za rad s bazama podataka, a na raspolaganju su mnogi, među kojima se ističe MySQL Workbench. Radi se o vizualnom okruženju opremljenom alatima za rad sa MySQL DBMS-om, a na raspolaganju je u dvije kategorije: Community Edition kao besplatna verzija te komercijalne verzija proširena dodatnim alatima i mogućnostima (Standard Edition ili Enterprise Edition). Od ostalih besplatnih i komercijalnih „front-end“ grafičkih alata za rad s MySQL RDBMS-om mogu se istaknuti phpMyAdmin, SQLBuddy, Adminer, HeidiSQL, Microsoft Access, DBStudio, Oracle SQL Developer i brojni drugi.

### 14.4.2 MySQL instalacija

Vrlo iscrpna službena podrška za MySQL RDBMS, kao i za Workbench GUI nalazi se na službenoj web lokaciji <http://www.mysql.com/>, a na istom mjestu moguće je pronaći i odgovarajuće instalacijske pakete. Za potrebe laboratorijskih uvjeta potrebno je preuzeti trenutno važeću verziju MySQL Community Edition na lokaciji <http://dev.mysql.com/downloads/mysql/>

**MySQL Community Server 5.6.22**

Select Platform:  
Microsoft Windows

Recommended Download:

**MySQL Installer 5.6**  
for Windows

All MySQL Products. For All Windows Platforms.  
In One Package.

Starting with MySQL 5.6 the MySQL Installer package replaces the server-only MSI packages.

Windows (x86, 64-bit), MySQL Installer MSI [Download](#)

Other Downloads:

Windows (x86, 32-bit), ZIP Archive (mysql-5.6.22-win32.zip)	5.6.22	342.0M	<a href="#">Download</a>
		MD5: 00abc99a71708d372ff073f870deabd   <a href="#">Signature</a>	
Windows (x86, 64-bit), ZIP Archive (mysql-5.6.22-winx64.zip)	5.6.22	347.5M	<a href="#">Download</a>
		MD5: 8810b875ff1651e3c91473faa7ed6509   <a href="#">Signature</a>	

Web obrazac za preuzimanje MySQL instalacijskog paketa

Instalacija započinje uobičajenim pokretanjem prethodno preuzetog instalacijskog paketa pri čemu korisnik prolazi kroz nekoliko dijaloških formi, od početnog odabira proizvoda i potvrde uvjeta licence, preko ostalih konfiguracijskih formi. U prvom prozoru nakon pokretanja instalacijske datoteke potrebno je kliknuti na dugme Add te u sljedećem dijalogu potvrditi uvjete licence, nakon čega se pojavljuje početni prozor instalacijskog procesa. U prvom dijelu instalacijskog procesa odabiru se programske komponente za instalaciju, od samog MySQL Server-a, preko MySQL Workbench vizualnog okruženja i drugih pomoćnih aplikacija, upravljačkih programa za povezivanje s vanjskim aplikacijama te programske dokumentacije. Na slici „Odabir instalacijskih komponenti“ odabiru se željene komponente, među kojima je potrebno odabrati MySQL Server komponentu u 32bit ili 64bit verziji, te MySQL Workbench alat kojega je po potrebi moguće dodati i naknadnom instalacijom. U sljedećem koraku dolazi do provjere programskih zahtjeva za odabranim komponentama te se klikom na dugme Execute automatizirano rješavaju nadopune sustava do funkcionalnosti pune podrške za instalaciju odabranih komponenti. U zadanom primjeru postupka instalacije pojavila se potreba za nadopunom Visual C++ 2013 Runtime sistemske komponente, koja se dodaje klikom na dugme Execute. Tada započinje odvojen postupak instalacije navedene komponente koji nakon završetka zahtijeva prepravak postupka prethodno započete instalacije, što se odnosi na ponovnu provjeru sistemskih uvjeta. Za nastavak instalacije potrebno je pritisnuti dugme Repair, nakon kojega se ponovno pojavljuje prozor MySQL instalacije gdje se istaknuta nedostajuća komponenta označava kao prisutna te se postupak može nastaviti klikom na dugme Next. Potrebno je naglasiti kako je moguće da ovaj izvojeni korak instalacije nedostajuće sistemske komponente neće morati uvijek obavljati jer ovisi o stanju sustava na kojemu se

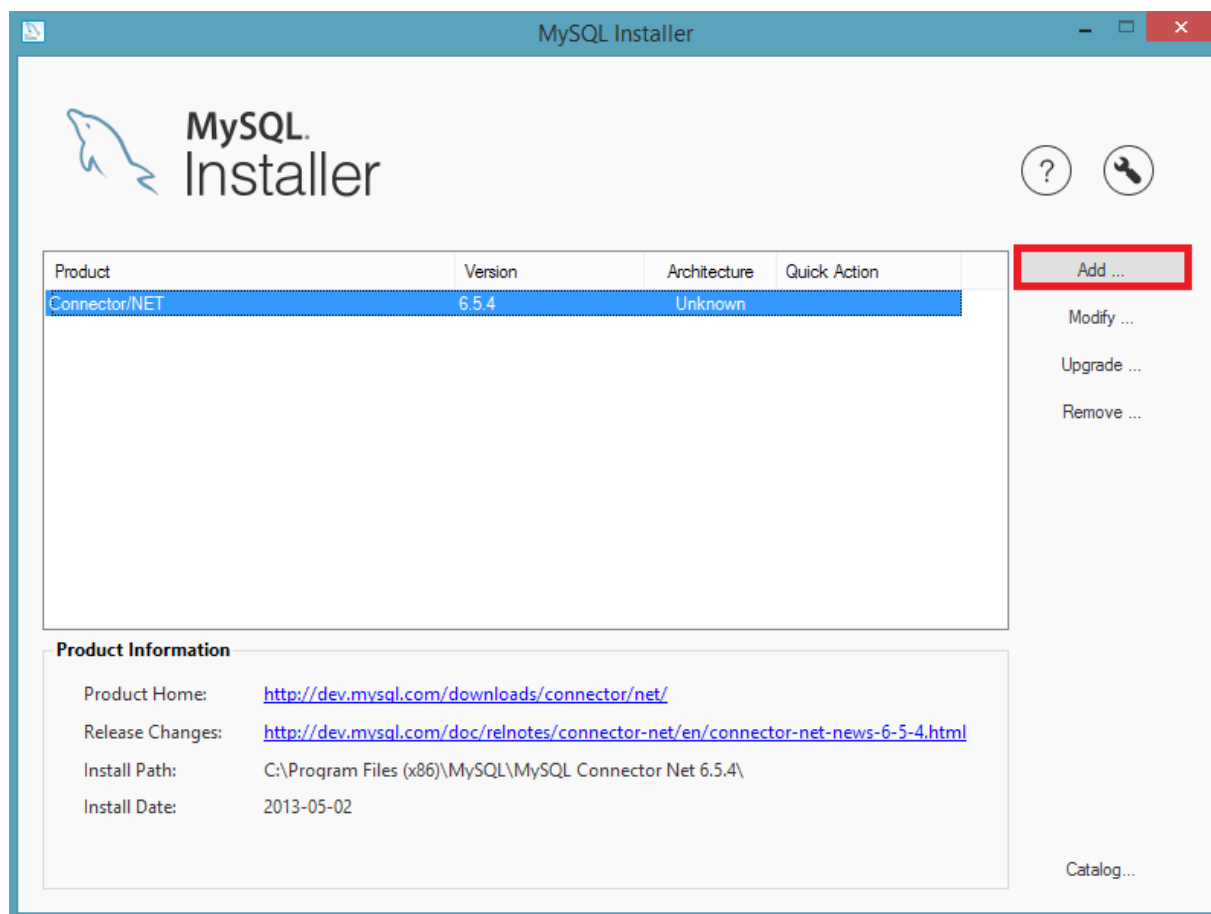
provodi instalacija. U tom slučaju, neće biti moguće odabrati dugme Execute, već će biti dostupan sljedeći korak nakon klika na dugme Next. U sljedećem koraku objavljuje se popis odabranih komponenti čija se instalacija pokreće klikom na dugme Execute. Nakon završene instalacije pokazuje se izvještaj o uspješno dovršenoj instalaciji odabranih komponenata te se klikom na dugme Next prelazi na fazu konfiguriranja svake od instaliranih komponenata. Prvo započinje konfiguracija SQL Server komponente te se klikom na dugme Next pojavljuje prozor Type and Networking. U sklopu ovog konfiguracijskog koraka odabire se vrsta primjene (Config Type) MySQL poslužitelja, a ponuđene su sljedeće mogućnosti: Development Machine, Server Machine, Dedicated Machine. Prva opcija namijenjena je računalima s razvojnim okruženjem koja sadrže i brojne druge pomoćne aplikacije te je stoga dodjela memorijskih resursa MySQL poslužiteljskom procesu minimizirana. Druga opcija namijenjena je poslužiteljskim računalima s više istovremenih poslužiteljskih procesa te je dodjela memorijskih resursa optimalno raspodijeljena među pojedinim procesima. U posljednjoj opciji maksimizirana je memorijska alokacija samo za potrebe jednog poslužiteljskog procesa, u ovom slučaju MySQL poslužitelja čime se ostvaruju pretpostavke za najviše performanse sustava. Za potrebe ove vježbe, dovoljno će biti odabrati opciju Development Machine. U sekciji mrežne povezivosti (Connectivity) određuju se TCP/IP parametri poput broja TCP vrata, što se u slučaju ovog instalacijskog postupka dodjeljuje automatski, iako je moguć i proizvoljan odabir slobodnih TCP vrata. Osim toga, potrebno je potvrdno označiti kontrolu *Open Firewall port for network access* radi omogućavanja pristupa poslužitelju klijentskim aplikacijama. U sljedećem koraku konfiguracije (Accounts and Roles) slijedi izrada korisničkog računa te korisničkih računa i njihovih privilegija. Klikom na dugme Add User otvara se dijaloški okvir za unošenje parametara novog korisnika. Pri tome je potrebno definirati korisničko ime (username), naziv korisničkog računala (localhost – računalo domaćin), korisnička uloga (Role) te lozinka (password). Za odabir korisničke uloge na raspolaganju su sljedeće:

- Backup Admin – ima prava za izvršenje sigurnosne pohrane baza
- DB Admin – administrator baza podataka s dozvolom izvršavanja svih zadaća
- DB Designer – dozvola za kreiranje sheme baze podataka i reverzni inženjering
- DB Manager – dozvola pristupa svim bazama podataka u sustavu
- Instance Manager – dodjeljuje prava za održavanje poslužitelja
- Monitor Admin – skup dozvola za nadzor rada poslužitelja
- Process Admin – dozvola za pristup, nadzor i gašenje svih korisničkih procesa na poslužitelju
- Replication Admin – dozvole za prava pristupa procesu replikacije
- Security Admin – dozvole za upravljanje pravima pristupa poslužitelju
- User Admin – dozvola za prava upravljanja korisničkim računima

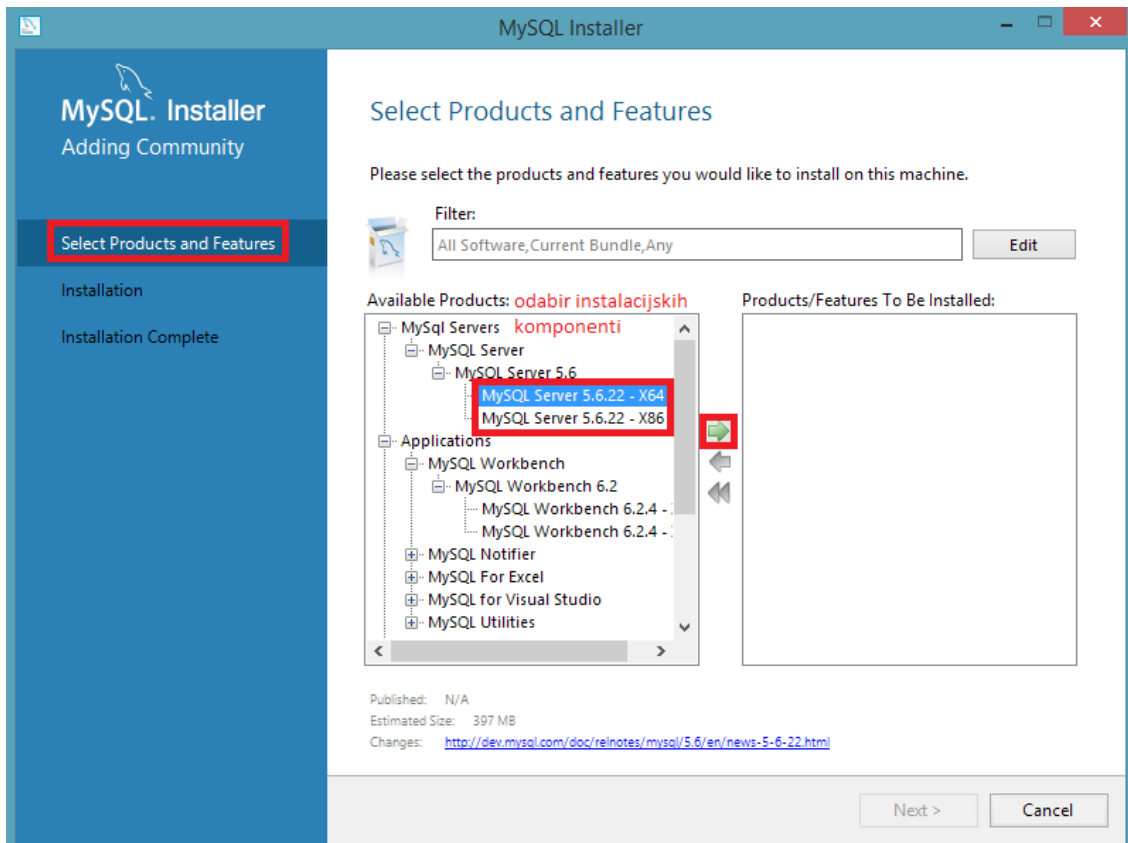
Nakon odabira odgovarajućeg tipa korisničke uloge te unosa ostalih podataka, potvrđuje se unijeto klikom na dugme OK. Tada se novokreirani korisnik pojavljuje u listi korisnika glavnog

instalacijskog prozora u koraku *Accounts and Roles*. U ovom trenutku moguće je dodavati još korisnika ili krenuti dalje klikom na dugme Next. U sljedećem koraku razrađuju se detalji postavki Windows usluge vezane za MySQL poslužitelj. Pri tom je moguće dodijeliti proizvoljan naziv usluge kao i način pokretanja u okolini operativnog sustava. Po završetku konfiguracije MySQL poslužitelja potrebno je kliknuti na dugme Finish te pristupiti krajnjem prozoru u kojem se još jednom potvrđuje postupak instalacije klikom na dugme Finish. Osim toga, moguće je odabrati želi li se odmah nakon potvrde pokrenuti MySQL Workbench alat. Konačno stanje predinstalacijske platforme, tzv. početnog prozora instalacije, prikazano je slikom „Lista instaliranih komponenata“. Na slici je vidljiv popis svih instaliranih komponenti te njihovih trenutnih verzija te akcijskih poveznica za dodatne konfiguracije u kolumni Quick Action, poput akcije Reconfigure koja se odnosi na rekonfiguraciju postavki MySQL poslužitelja.

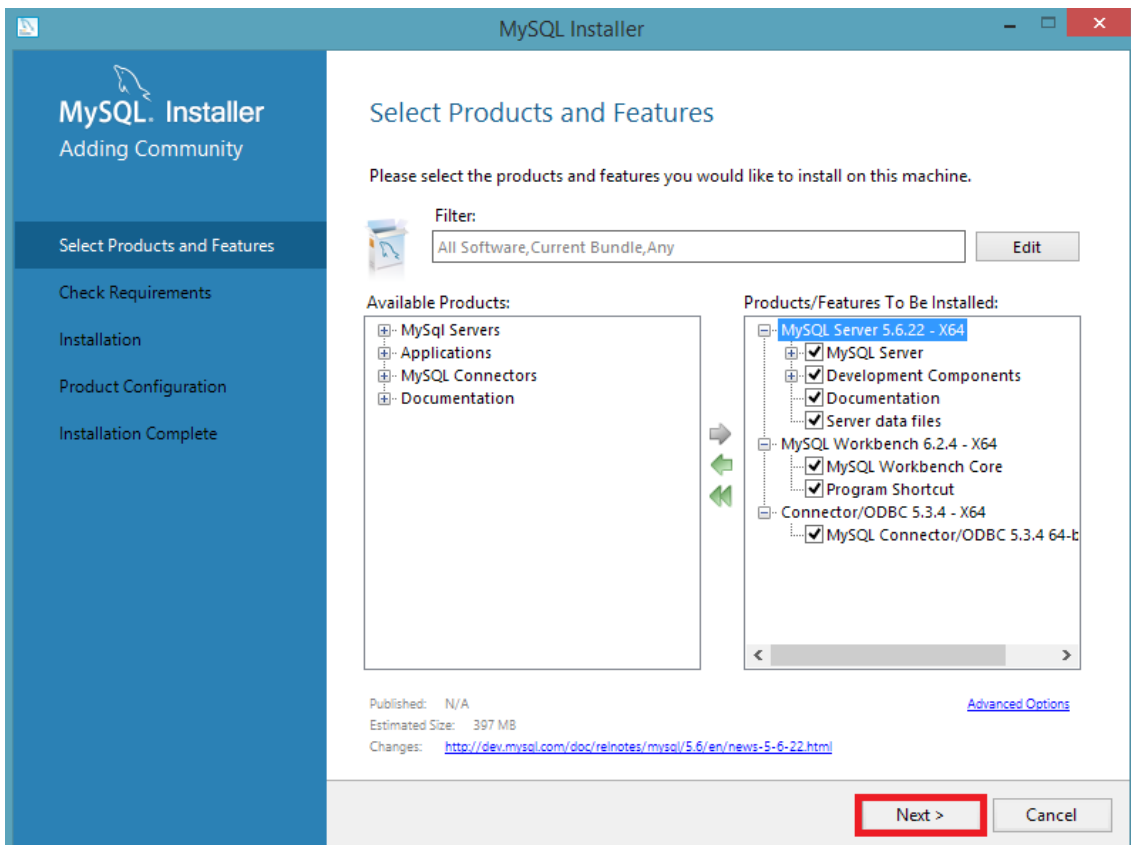
Početni korak instalacije:



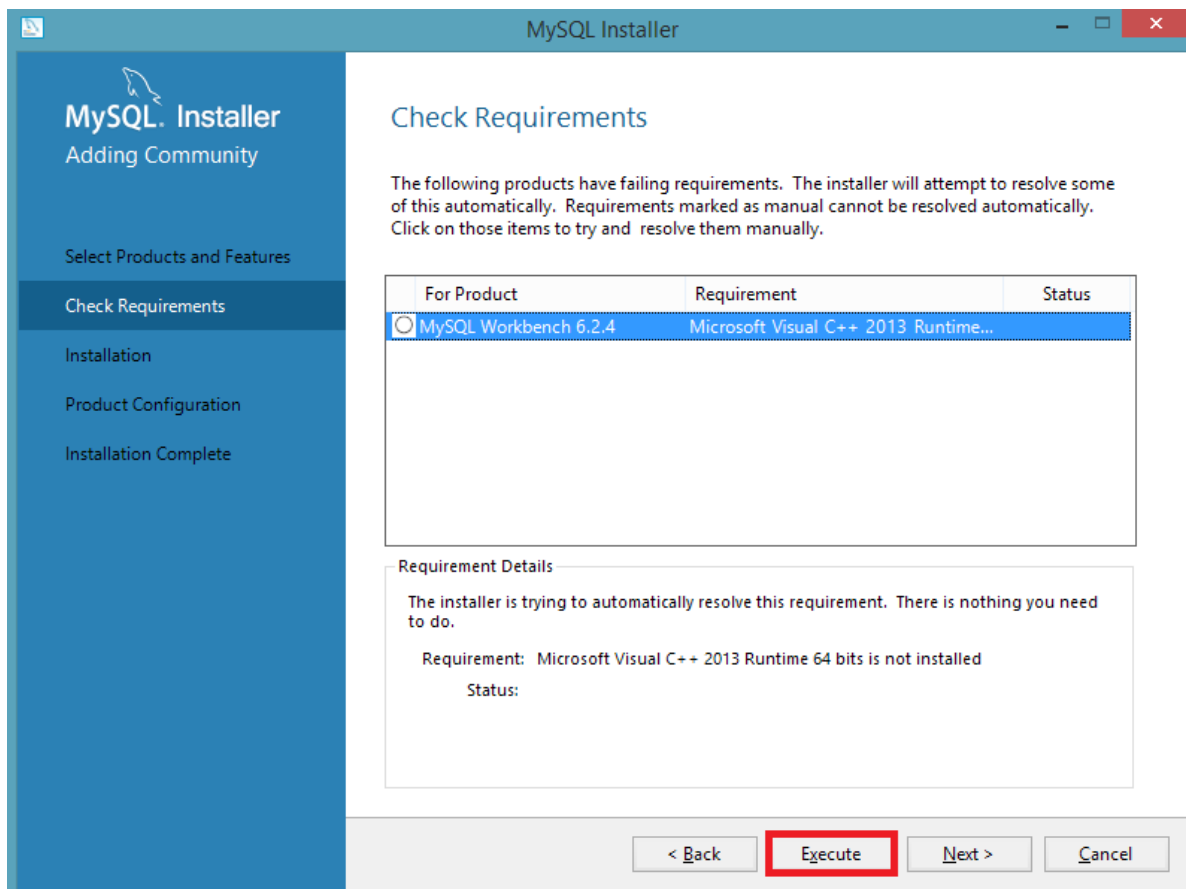
Odabir instalacijskih komponenti:



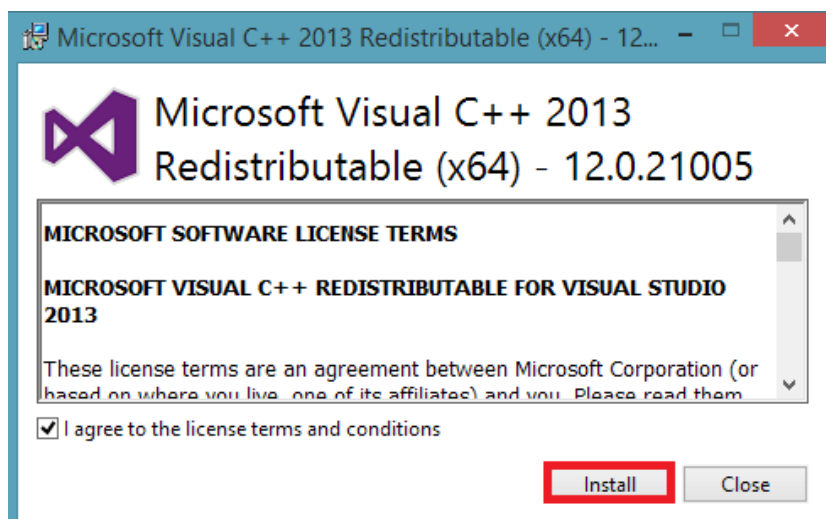
Odabrane instalacijske komponente:



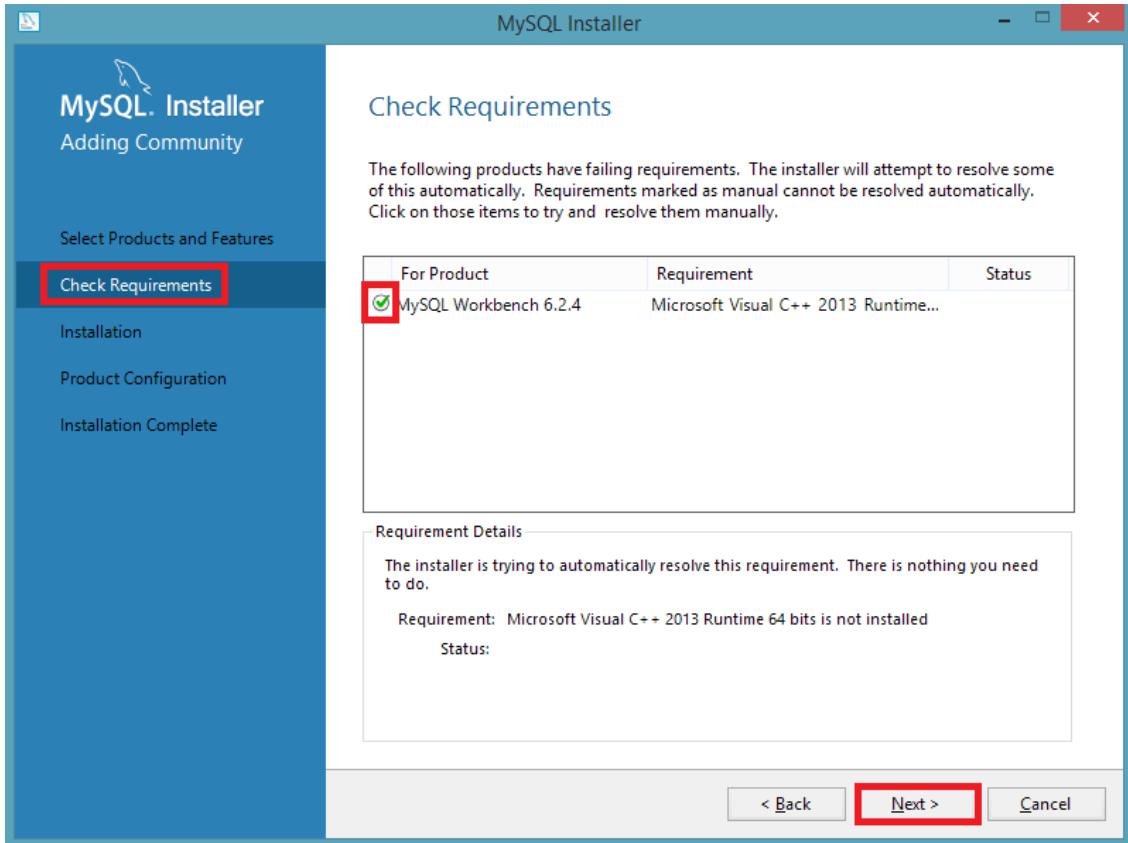
Korak provjere zahtjeva za podršku sustava instalaciji odabranih komponenata:



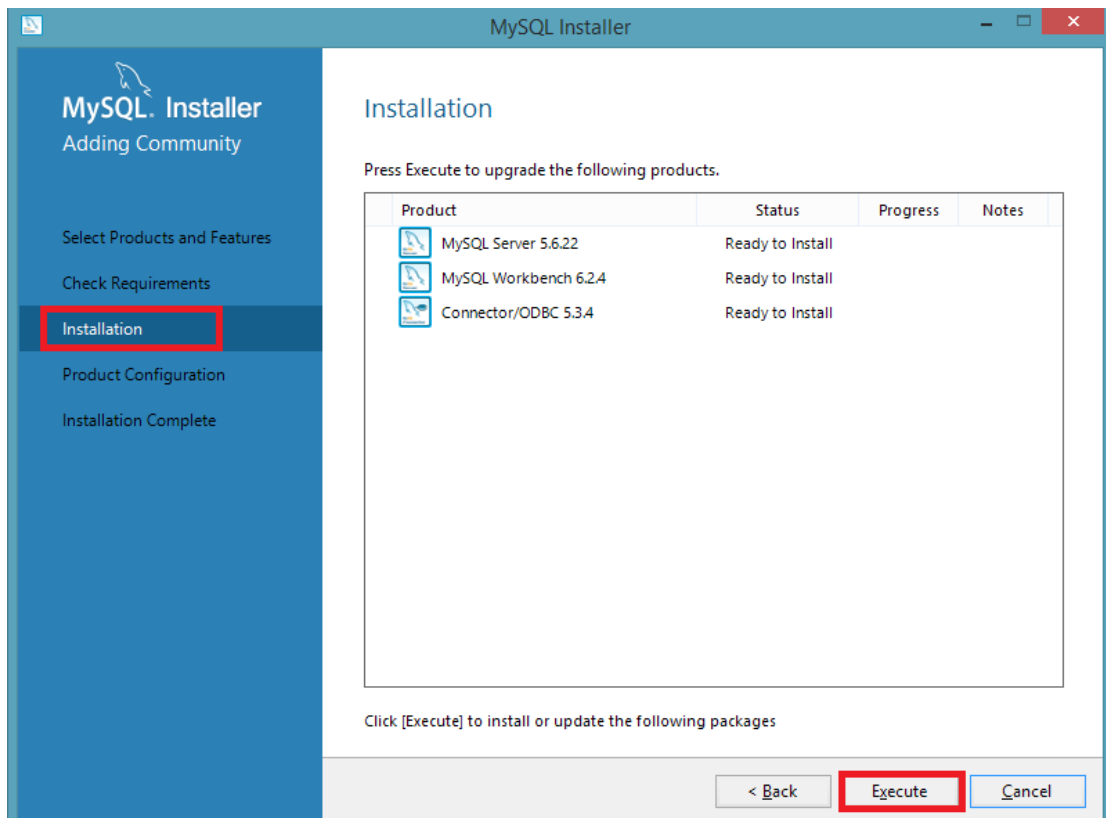
Instalacije sistemske komponente za potporu instalaciji odabranih komponenata:



Uspješna provjera instalacijskih zahtjeva i prelazak na sljedeći korak:

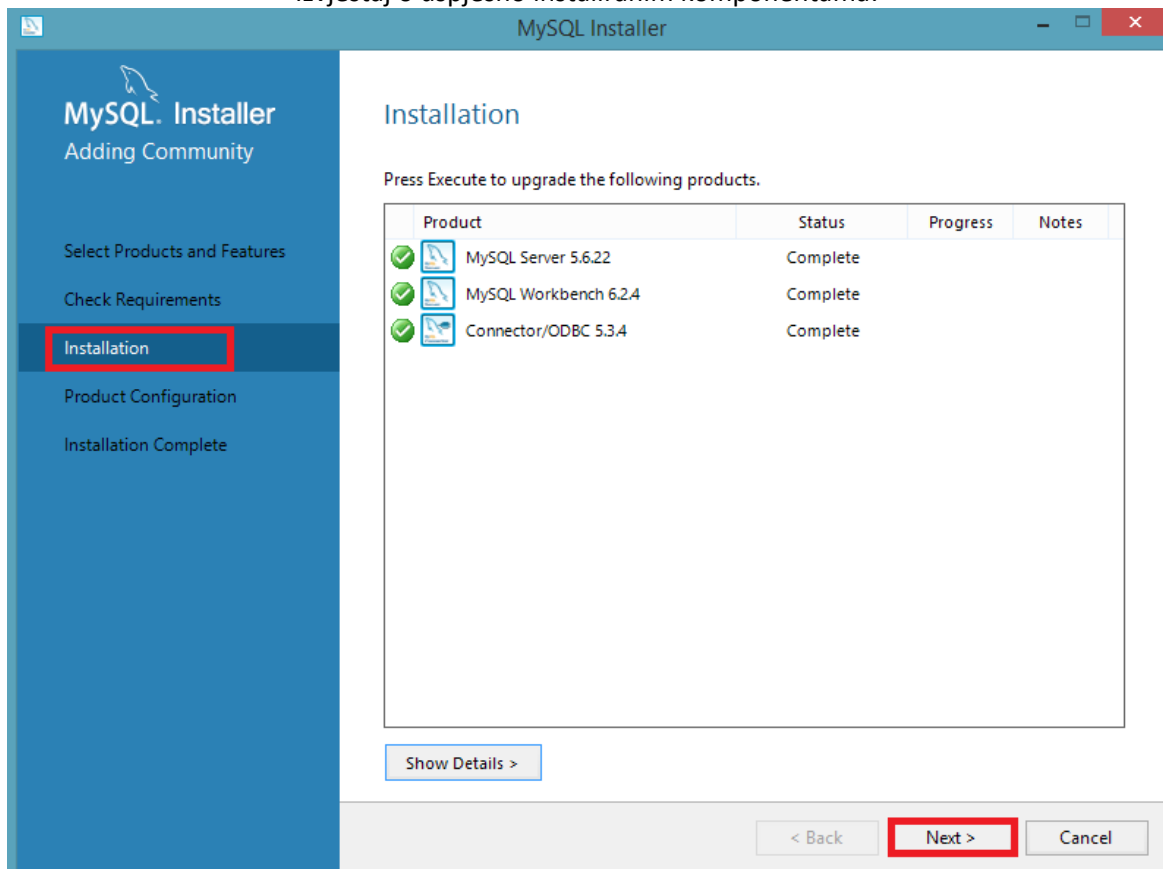


Potvrda početka instalacije odabranih komponenata:

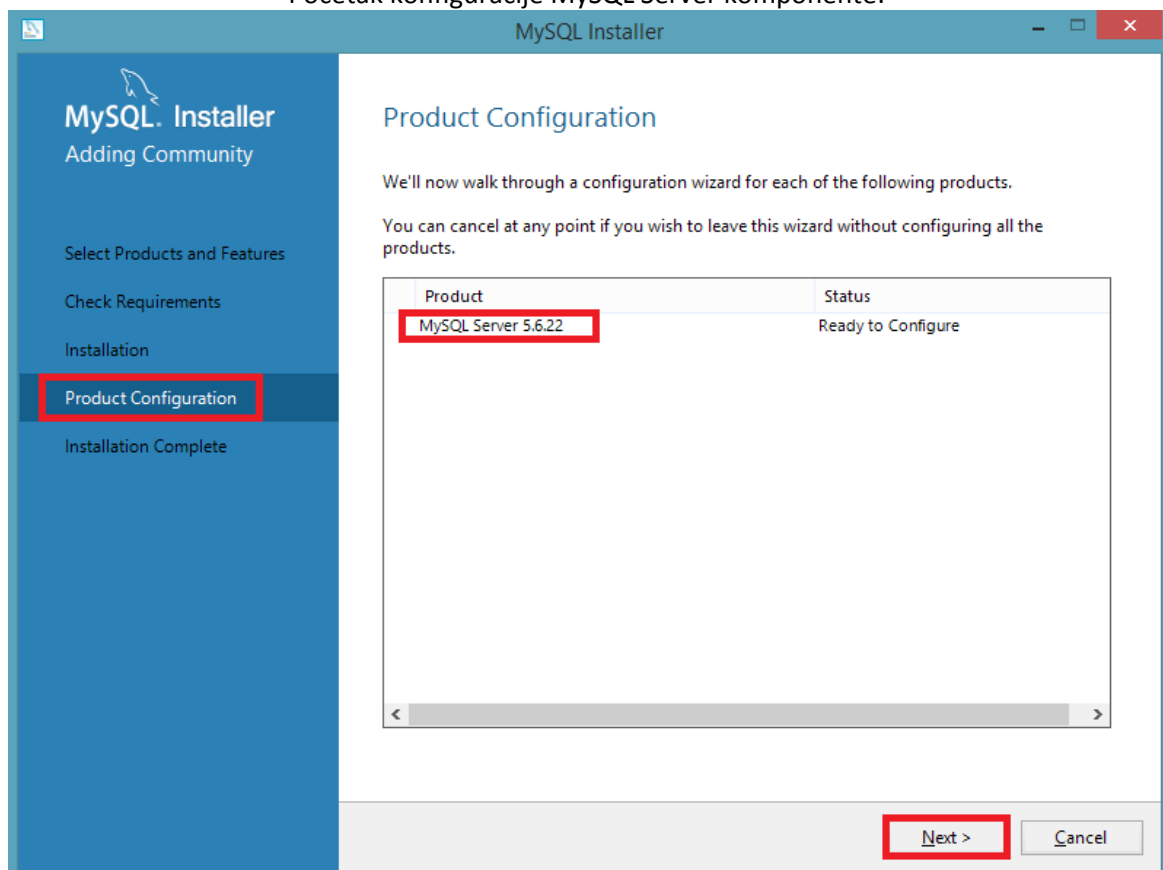




## Izveštaj o uspješno instaliranim komponentama:



## Početak konfiguracije MySQL Server komponente:



## Mrežne postavke MySQL poslužitelja

MySQL Installer  
MySQL Server 5.6.22

Type and Networking

Accounts and Roles

Windows Service

Apply Server Configuration

### Type and Networking

Server Configuration Type

Choose the correct server configuration type for this MySQL Server installation. This setting will define how much system resources are assigned to the MySQL Server instance.

Config Type: **Development Machine**

Connectivity

Use the following controls to select how you would like to connect to this server.

TCP/IP Port Number: 3306

Open Firewall port for network access

Named Pipe Pipe Name: MYSQL

Shared Memory Memory Name: MYSQL

Advanced Configuration

Select the checkbox below to get additional configuration page where you can set advanced options for this server instance.

Show Advanced Options

Next > Cancel

## Postave korisničkih računa i privilegija

MySQL Installer  
MySQL Server 5.6.22

Type and Networking

**Accounts and Roles**

Windows Service

Apply Server Configuration

### Accounts and Roles

Root Account Password

Enter the password for the root account. Please remember to store this password in a secure place.

MySQL Root Password:

Repeat Password:

Password minimum length: 4

MySQL User Accounts

Create MySQL user accounts for your users and applications. Assign a role to the user that consists of a set of privileges.

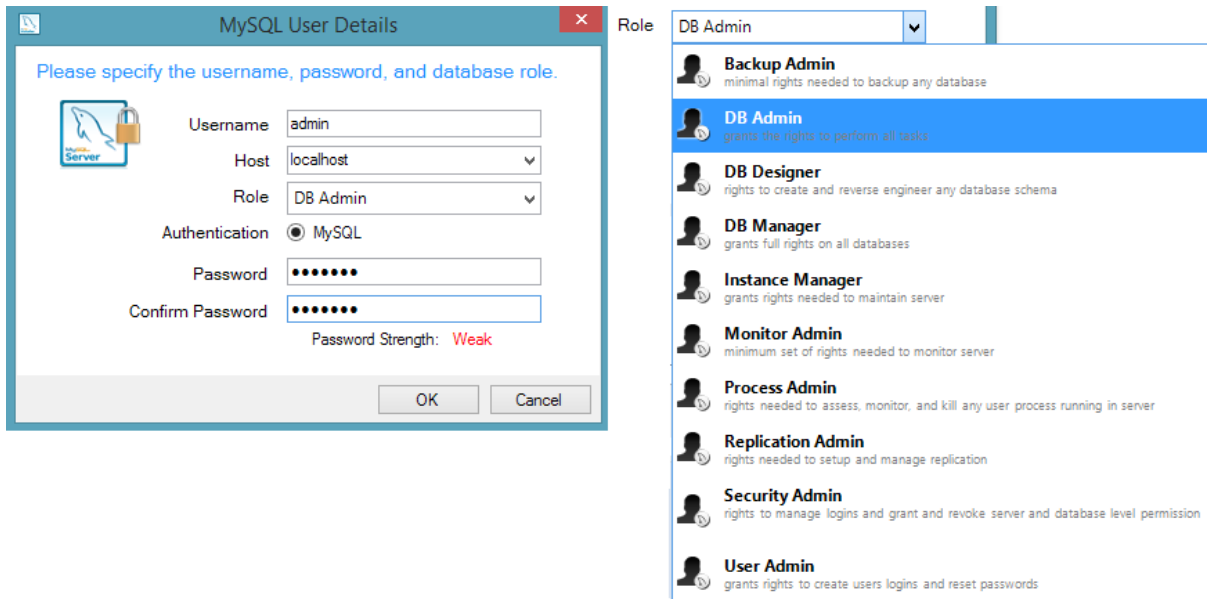
MySQL Username	Host	User Role
----------------	------	-----------

Add User

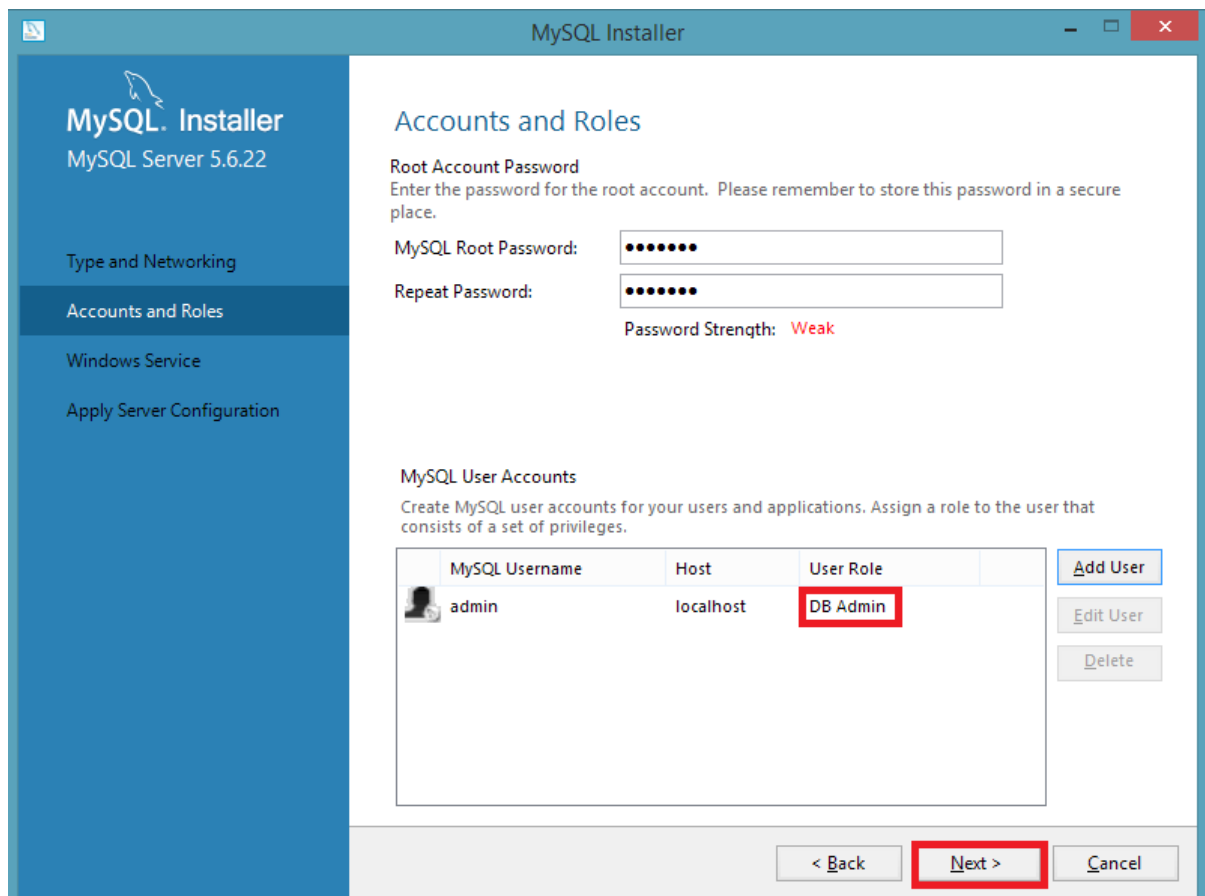
Edit User

Delete

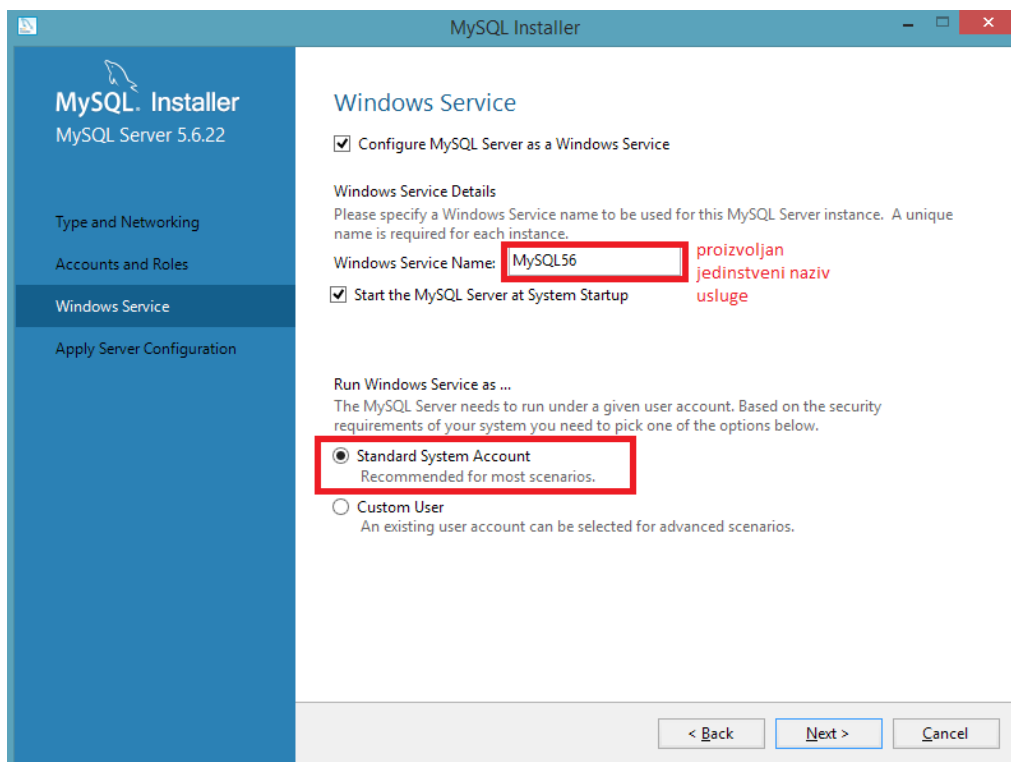
< Back Next > Cancel



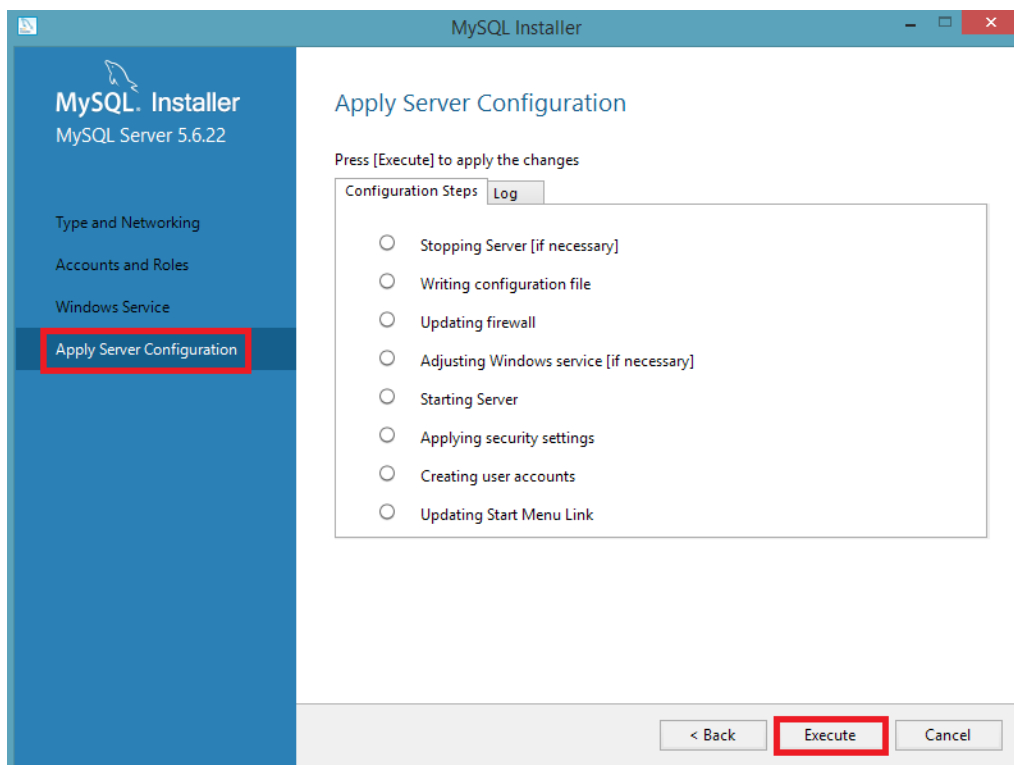
Kreiranje korisničkog računa i popis korisničkih uloga



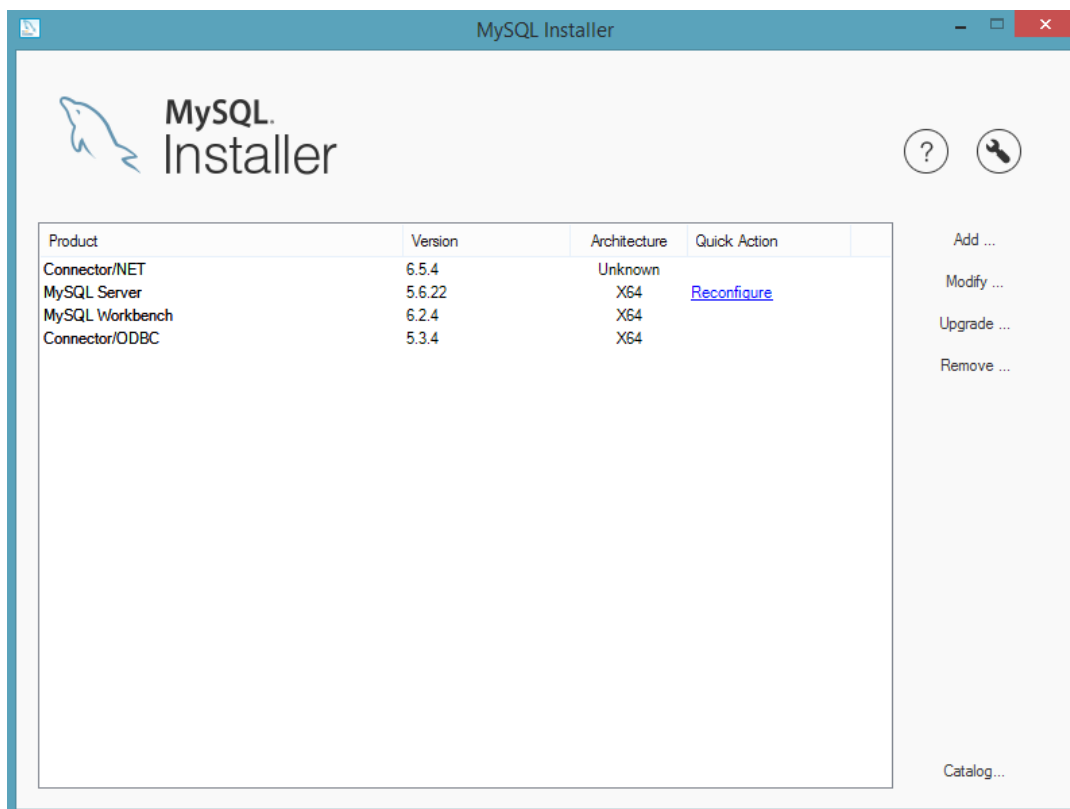
Kreirani administratorski korisnički račun



Konfiguracija MySQL Windows procesa



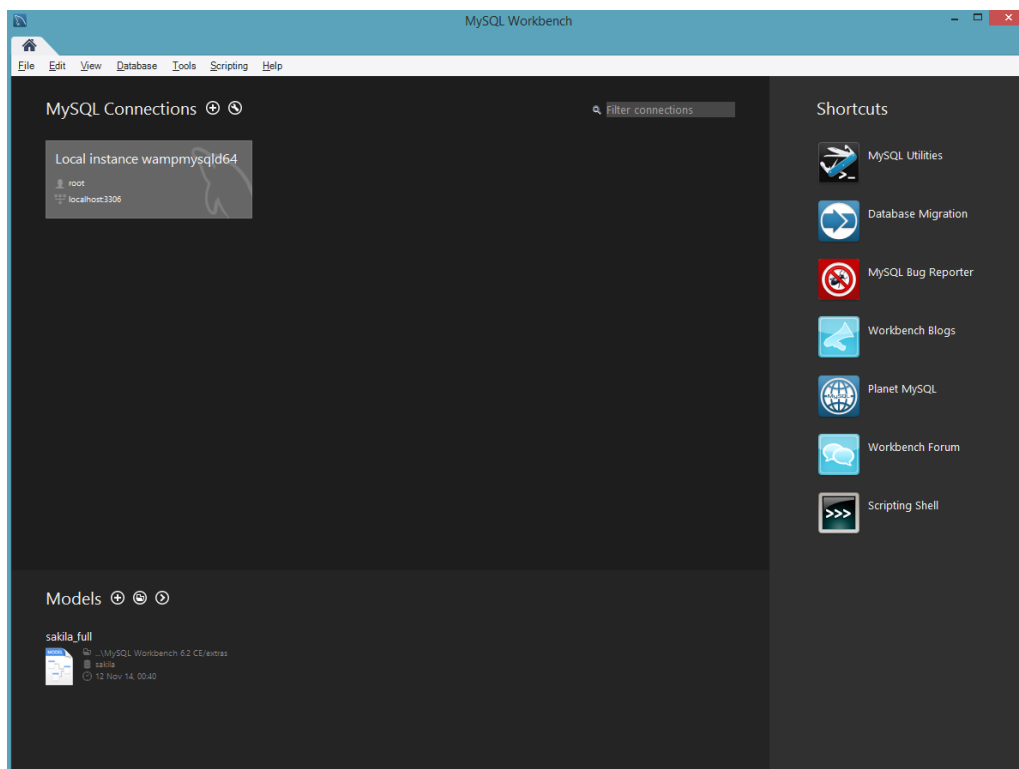
Potvrda konfiguracije MySQL poslužitelja



Lista instaliranih komponenta

### 14.4.3 MySQL Workbench

MySQL Workbench dostupan je u tri izdanja, ovisno o uvjetima korištenja: besplatna ili tzv. Community inačica, te dvije komercijalne: Standard i Enterprise. Instalacija MySQL Workbench Community inačice dostupna je kao samostalni instalacijski paket na službenim stranicama distributora, ili kao dio MySQL Server instalacijskog paketa te je za potrebe ove vježbe instalirana u sklopu prethodnih uputa. Odabirom automatskog pokretanja MySQL Workbench alata u posljednjem koraku instalacije otvara se glavna upravljačka ploča prikazana sljedećom slikom.

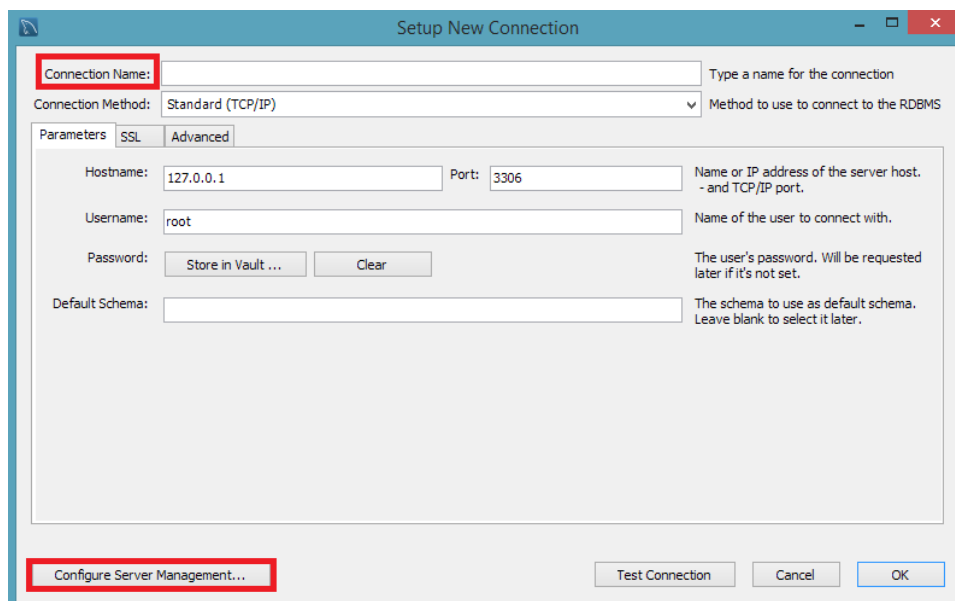


Početni prozor MySQL Workbench 6 CE grafičkog alata

Početni prozor MySQL Workbench alata podijeljen je u tri glavne sekcije:

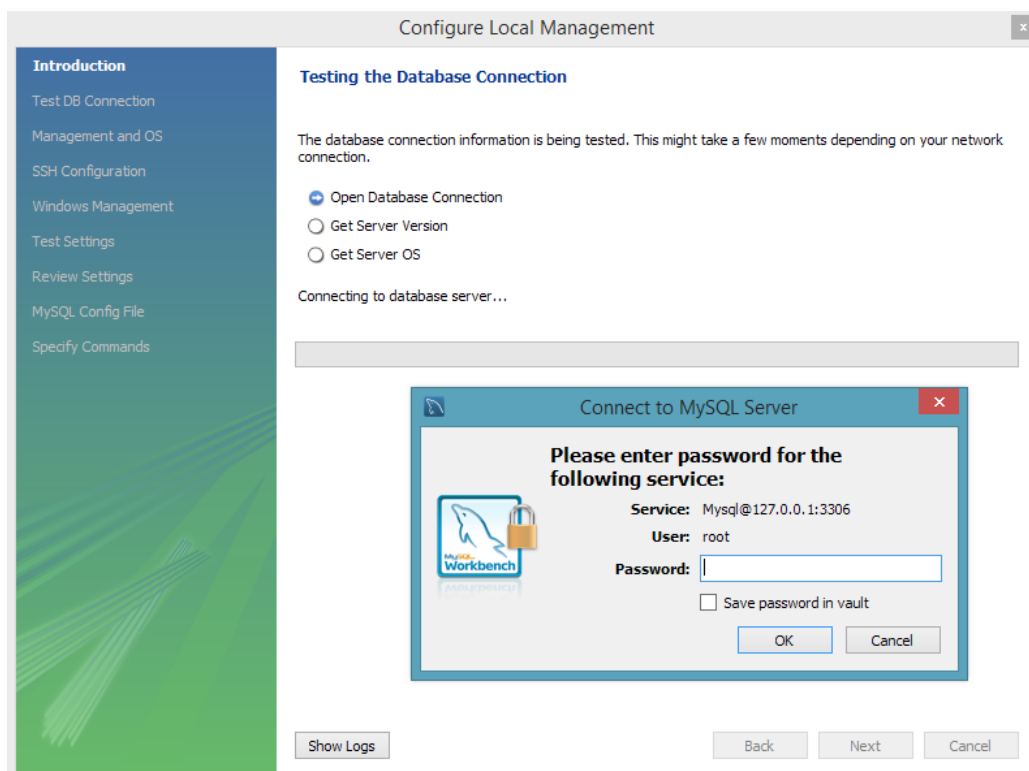
- MySQL Connections – lista konekcija na raspoložive instance MySQL poslužitelja
- Models – lista posljednje korištenih modela
- Shortcuts – kratice za najčešće funkcije ili alate

U sekciji MySQL konekcija inicijalno se nalazi predefinirana konekcija nastala tijekom instalacije MySQL poslužitelja. Osim ponuđenih, prethodno kreiranih konekcija, moguće je kreirati i nove konekcije klikom na dugme „+“ iza naziva „MySQL Connections“. Tada se otvara prozor obrasca za podešavanje parametara nove konekcije u kojemu se definiraju naziv konekcije, TCP/IP parametri i ostali parametri bitni za uspostavljanje konekcije s lokalnim ili udaljenim MySQL poslužiteljem. Kreiranje nove konekcije moguće je ostvariti iz aspekta početnog panela MySQL Workbench alata, klikom na ikonu „+“ u sekciji MySQL Connections. Tada se otvara početni prozor čarobnjaka za unos postavki nove konekcije. Postavka nove konekcije započinje dodjeljivanjem proizvoljnog naziva konekcije te se klikom na dugme Configure Server Management... upućuje na detaljno definiranje parametara kroz navođeni slijed dijaloških prozora.



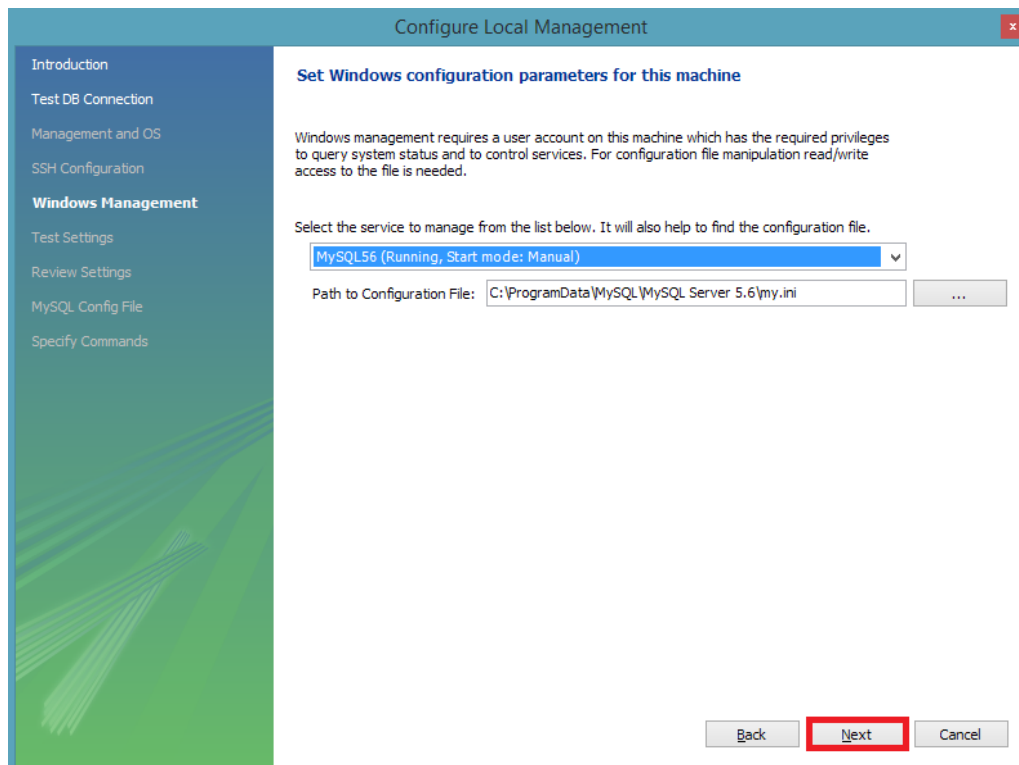
Početni prozor čarobnjaka za uspostavu nove MySQL konekcije

Sam početak konkretne konfiguracije započinje Configure Local Management prozorom gdje se navode uvodne napomene i informacije o postupku koji slijedi. Klikom na dugme Next započinje korak testiranja konekcije na instancu MySQL poslužitelja radi prikupljanja informacija o poslužitelju za potrebe daljnje konfiguracije. Pri tome je potrebno izvršiti autorizaciju korisnika unosom odgovarajuće lozinke, kreirane u prošlom odlomku, te kliknuti na dugme OK.



Korak provjere valjanosti MySQL konekcije te prikupljanja informacija o sustavu

U slučaju uspješne provjere valjanosti konekcije, postupak se nastavlja klikom na dugme Next gdje se odabire odgovarajuća MySQL Windows usluga (obično je nazvana prema trenutno instaliranoj inačici, pa je to u ovome slučaju MYSQL56). Osim toga navedena je putanja do konfiguracijske datoteke my.ini koja se može po volji mijenjati ili ostaviti kako je predložena.

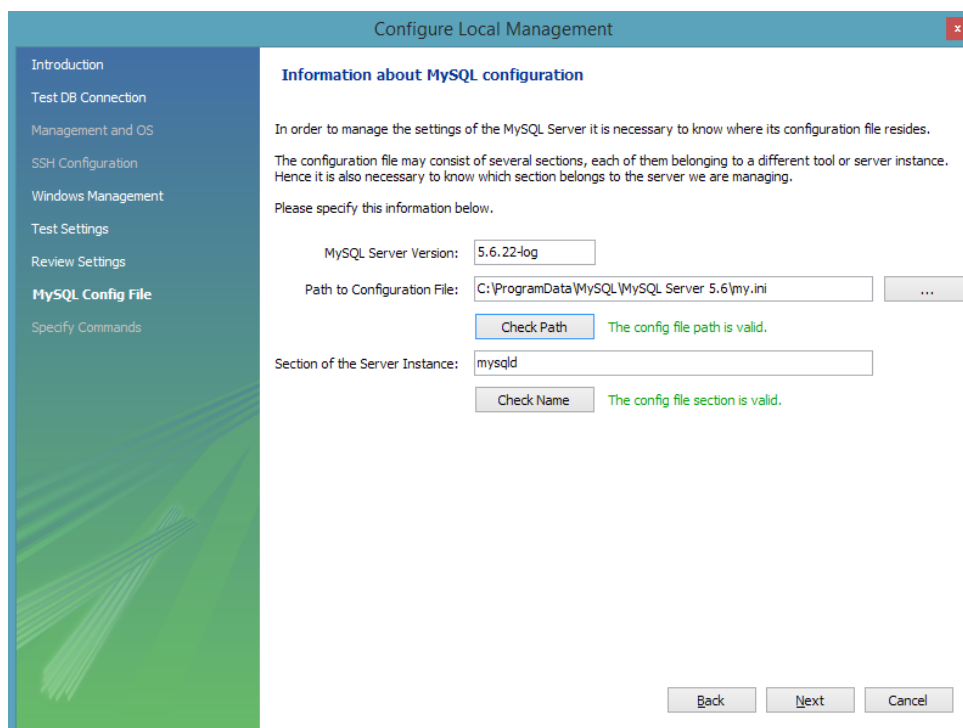


Konfiguracija Windows systemske usluge MySQL poslužitelja

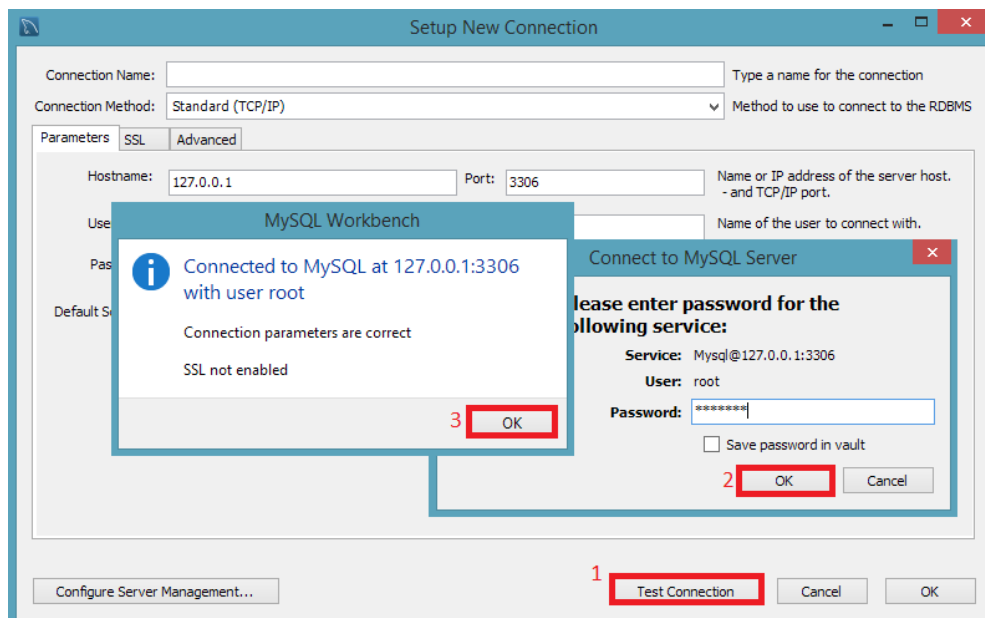
U nastavku se provjerava povezivost na računalo domaćina (tzv. host machine) te dostupnost i funkcionalnost kontrola za pokretanje i zaustavljanje Windows usluge MySQL poslužitelja kao i upotrebljivost (dozvola čitanja i pisanja) konfiguracijske datoteke my.ini na prethodno definiranoj lokaciji. Klikom na dugme Next otvara se prozor sa zahtjevom za pregled (*I'd like to review the settings again*) ili potvrdu prethodno unijetih postavki (Continue). U slučaju potvrde klikom na dugme Continue, postupak konfiguracije se završava te se upravo kreirana konekcija dodaje na listu postojećih konekcija u sekciji MySQL Connections na početnom panelu MySQL Workbench alata. U sklopu prozora za pregled dosadašnjih postavki moguće je i intervenirati na dodatnu izmjenu željenih postavki konfiguracijske datoteke MySQL poslužitelja, označavanjem kontrolne kućice Change Parameters. Ovisno o odabranom, moguće je kliknuti na dugme Finish ili Next. Klikom na dugme Next otvara se dodatni prozor s mogućnošću definiranja naziva MySQL poslužitelja, putanje do konfiguracijske datoteke my.ini te naziva sekcije poslužiteljske instance. U sljedećem prozoru, nakon klika na dugme Next, moguće je definirati vlastite komande za pokretanje i zaustavljanje usluge MySQL poslužitelja u Windows operacijskom sustavu, a u slučaju predpodešenih vrijednosti parametre je potrebno ostaviti prazne. Klikom na dugme Finish vraća se početni dijaloški okvir za konfiguraciju konekcije MySQL poslužitelja te je klikom na



dugme Test Connection moguće provjeriti jesu li svi parametri ispravno definirani i je li novokreirana MySQL konekcije funkcionalna.



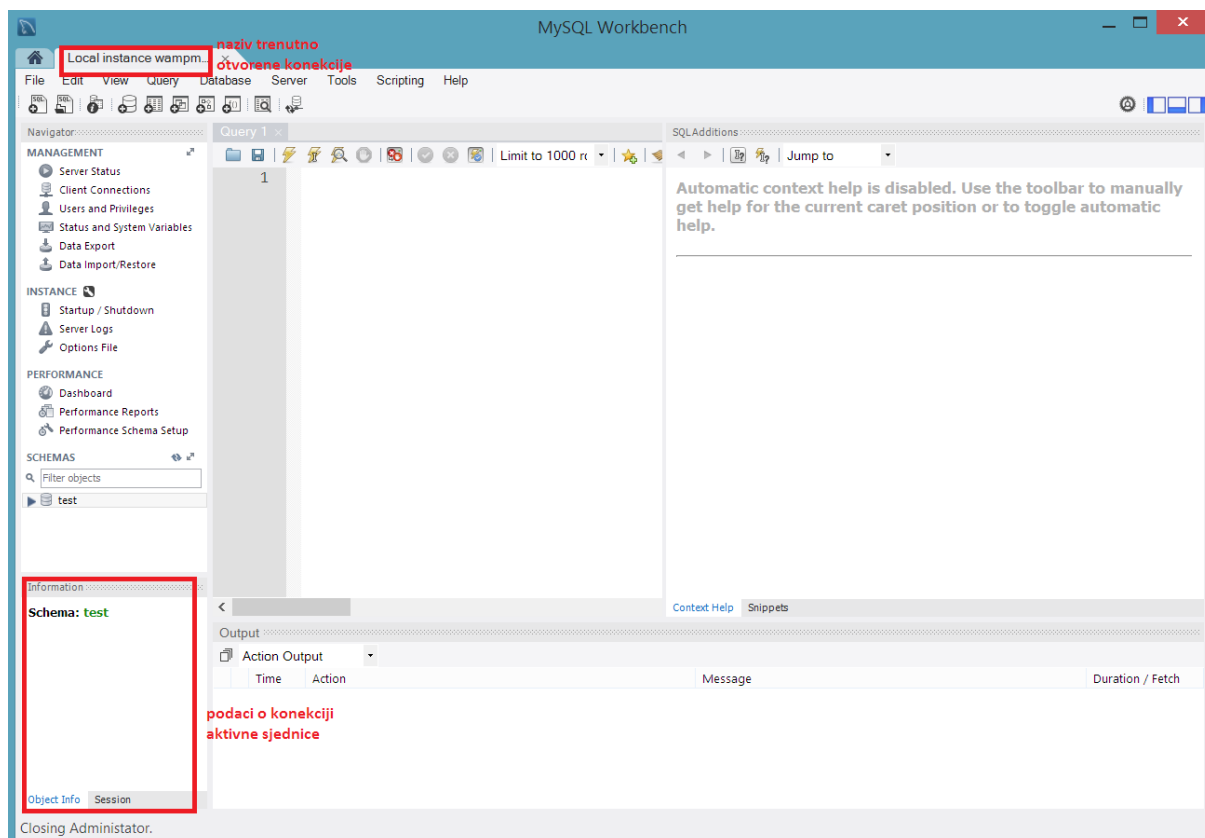
Korak promjene i provjere MySQL konfiguracijske datoteke



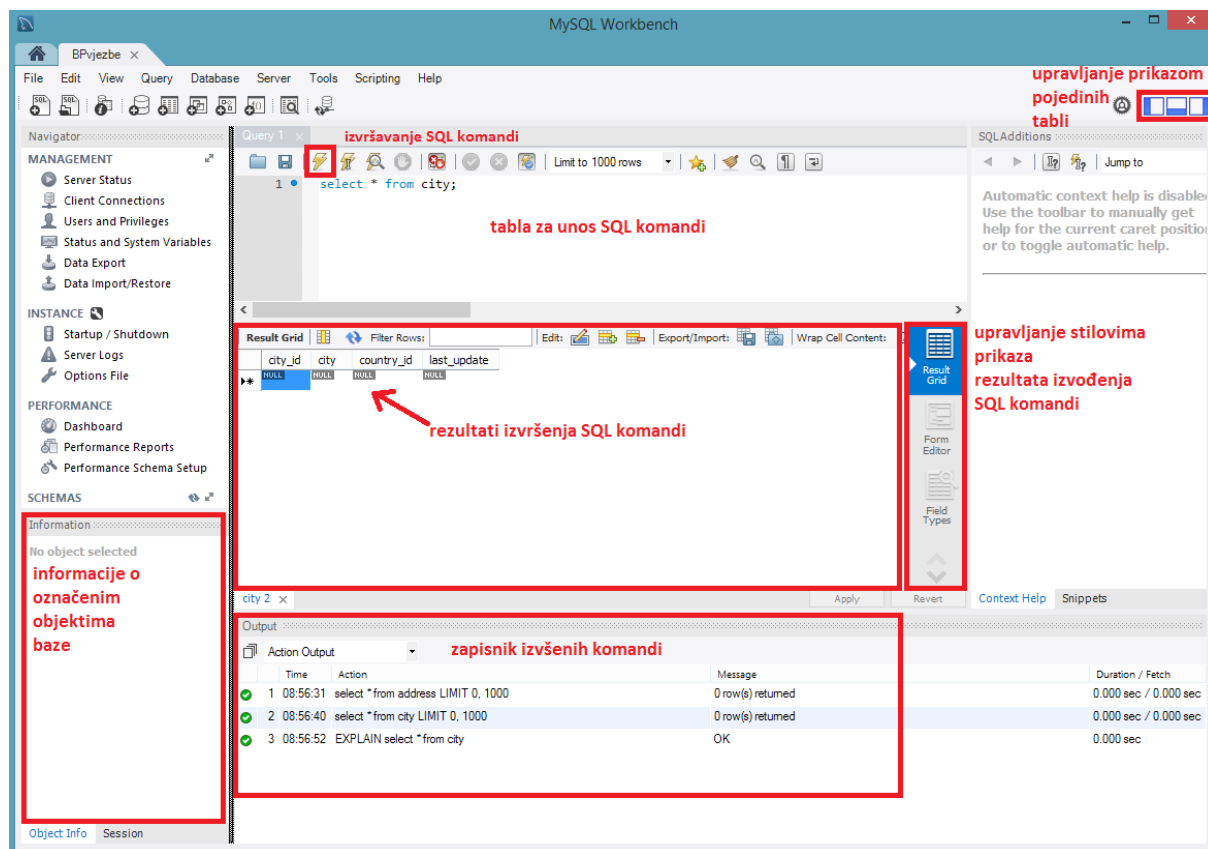
Konačna provjera funkcionalnosti nove MySQL konekcije

U sekciji Models MySQL Workbench početnog prozora nalazi se popis podatkovnih modela te je u postinstalacijskom početnom stanju priložen jedan ogledni primjer nazvan sakila. Također je moguće i kreiranje novih projekata, klikom na dugme „+“ iza naziva „Models“ čime se otvara razvojna okolina za logičko modeliranje podataka.





Za potrebe ovih vježbi prikazat će se dio MySQL Workbench programa koji služi za rad s bazama podataka, a temelji se na jeziku SQL. Radi se o vizualnom SQL editoru (Visual SQL Editor) kojim se omogućuje izrada, uređivanje i pokretanje SQL komandi.



SQL editor – alat za upravljanje SQL upitima

Na prethodnoj slici istaknuto je moguće područje za izravan unos SQL komandi, koje se potom izvršavaju klikom na ikonu „Izvršavanje SQL komandi“ smještenoj u alatnoj traci SQL editora. Osim navedene funkcije, na raspolaganju su još i funkcije spremanja upita, otvaranja, te cjelokupnog ili djelomičnog izvršavanja SQL skripti, izvršenja SQL skripti s funkcijom EXPLAIN koja vraća plan izvršenja zadanog upita te rezultatni skup, funkcije transakcija (Commit i Rollback), zaustavljanje izvršavanja skripte uz otvoren proces transakcije, uključivanje mogućnosti zaustavljanja izvršavanja skripte uslijed pogrešaka, uključivanje izravnog izvršavanja svih SQL izraza u skripti unutar iste transakcije (nezavisne SQL transakcije zahtijevaju zasebne MySQL konekcije), limitiranje dohvata entorki pojedinim SQL upitom na zadani broj, spremanje upita u listu favorita, prilagođeno formatiranje i poboljšano strukturiranje napisanog koda, aktivacija ugrađenog pretraživača pojmova i vrijednosti SQL editora, prikaz nevidljivih znakova poput uvlake, nove linije i sl. te posljednja funkcija prelamanja teksta.



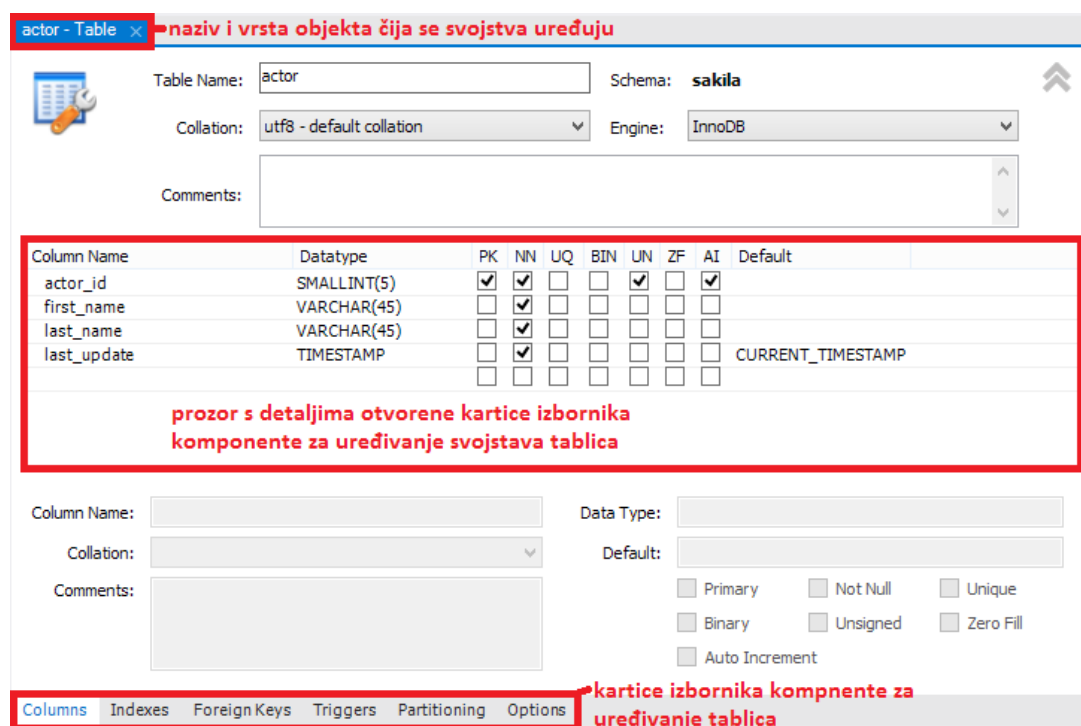
Izborničke opcije glavnog izbornika, Query i Edit sadrže dodatne kontrole i opcije povezane s prozorima SQL Editora. Tako se u izborniku Query nalaze komande poput Execute za izvršavanje dijela komandi bilo djelomičnih ili cjelovitih, zatim alati za grafičko objašnjenje zadanih SQL komandi, funkcije upravljanje izvršavanjem SQL skripti, limitiranja broja entorki dohvaćenih upitima, funkcije transakcija,

izvoz podataka rezultatnih skupova i dr. Podizbornik Format izbornika Edit sadrži funkcije oblikovanja prikaza SQL izraza poput upravljanja uvlakama, veličinom slova ključnih riječi, komentarima i dr.

Dio prozora koji se odnosi na prikaz rezultata izvođenja SQL komandi, Result Grid, obuhvaća funkcije poništavanja uvedenih sortiranja (Reset), obnavljanja prikaza rezultata ponovnim izvođenjem istih SQL komandi (Refresh), pretraživanja sadržaja rezultatnog skupa entorki (Filter Rows), uređivanja vrijednosti entorki (Edit Current Row), brisanja postojećih i dodavanja novih entorki (Delete Selected Rows, Add New Row), izvoza i uvoza podataka (Export, Import) te prelamanja sadržaja pojedinih ćelija.



Za potrebe ovih vježbi dobro je poznavati i komponentu za upravljanje tablicama u bazi podataka koja omogućuje izmjenu definicije postojećih tablica, kao i kreiranje novih uz pomoć grafičkih alata. Iako postoji nekoliko načina kako pristupiti navedenoj komponenti, najjednostavnije joj je pristupiti desnim klikom na naziv tablice u okviru sekcije Schemas, pod instancom željene baze podataka te odabirom funkcije ALTER TABLE. Pokretanjem ove funkcije otvorit će se novi početni prozor za uređivanje definicije odabrane tablice.

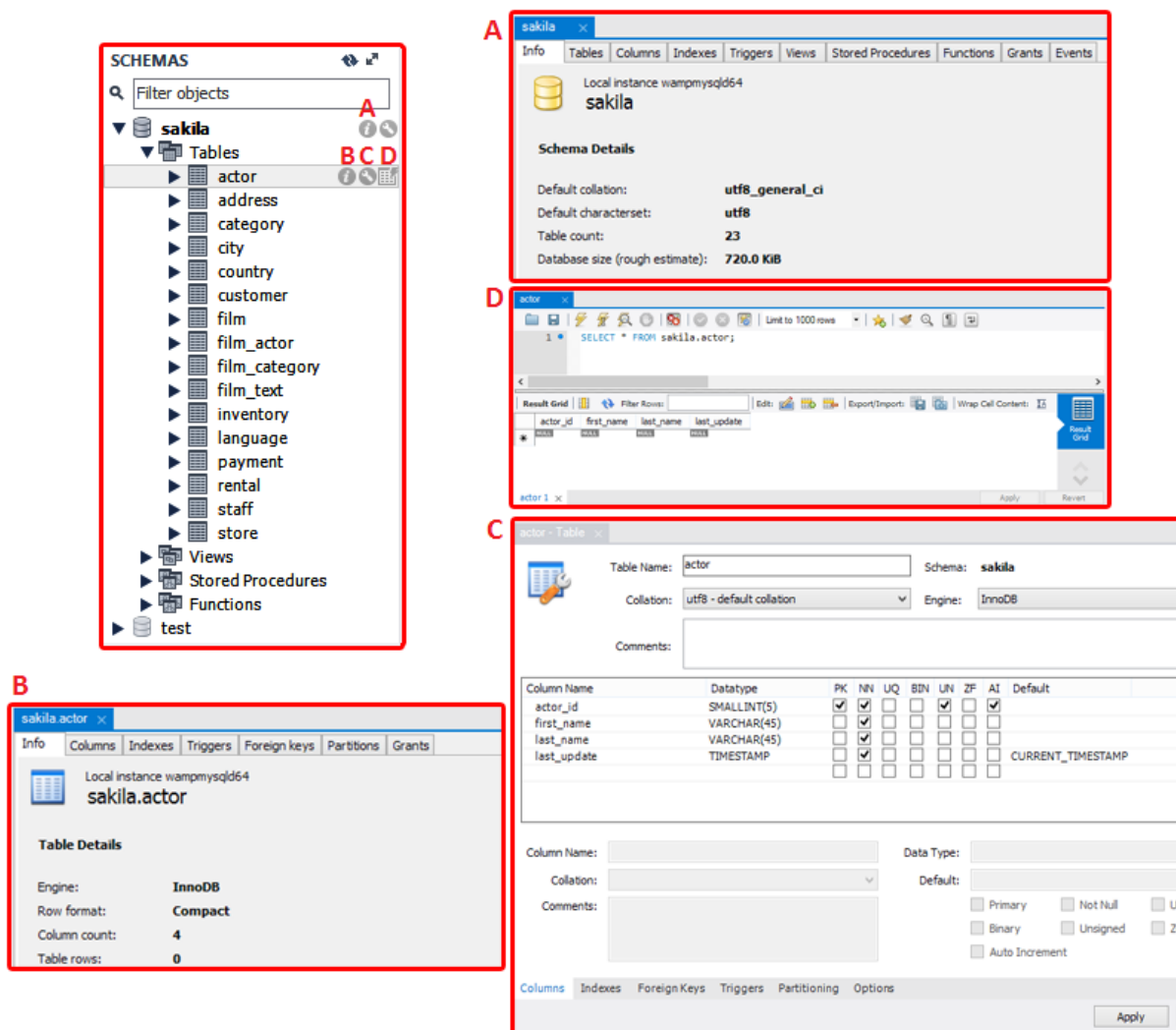


Prikaz komponente za uređivanje definicije tablice

Prozor na prethodnoj slici podijeljen je na dio s karticama izbornika (Columns, Indexes, Foreign Keys, Triggers, Partitioning, Options) te dio s pripadajućim detaljima pojedinih kartica. Tako je kartica Columns vezana za opcije i kontrole samih atributa, kartica Indexes za indekse i pripadajuća svojstva, kartica Foreign Keys za strane ključeve, kartica Triggers za definicije trigera, te kartice za svojstva particioniranja i ostalih općih opcija tablica (Partitioning, Options). Za potrebe ove vježbe bitno je

poznavati upravo opisanu komponentu i većinu mogućnosti koje su time dane na raspolaganje. Za samo izvođenje zadataka vježbe, neće biti potrebe koristiti se navedenim funkcijama, iako ih je općenito poželjno poznavati.

Pregledavanje dostupnih baza podataka i pripadajućih objekata te pristup funkcijama njihova dodavanje ili uređivanja dostupno je unutar okvira Object Browser.

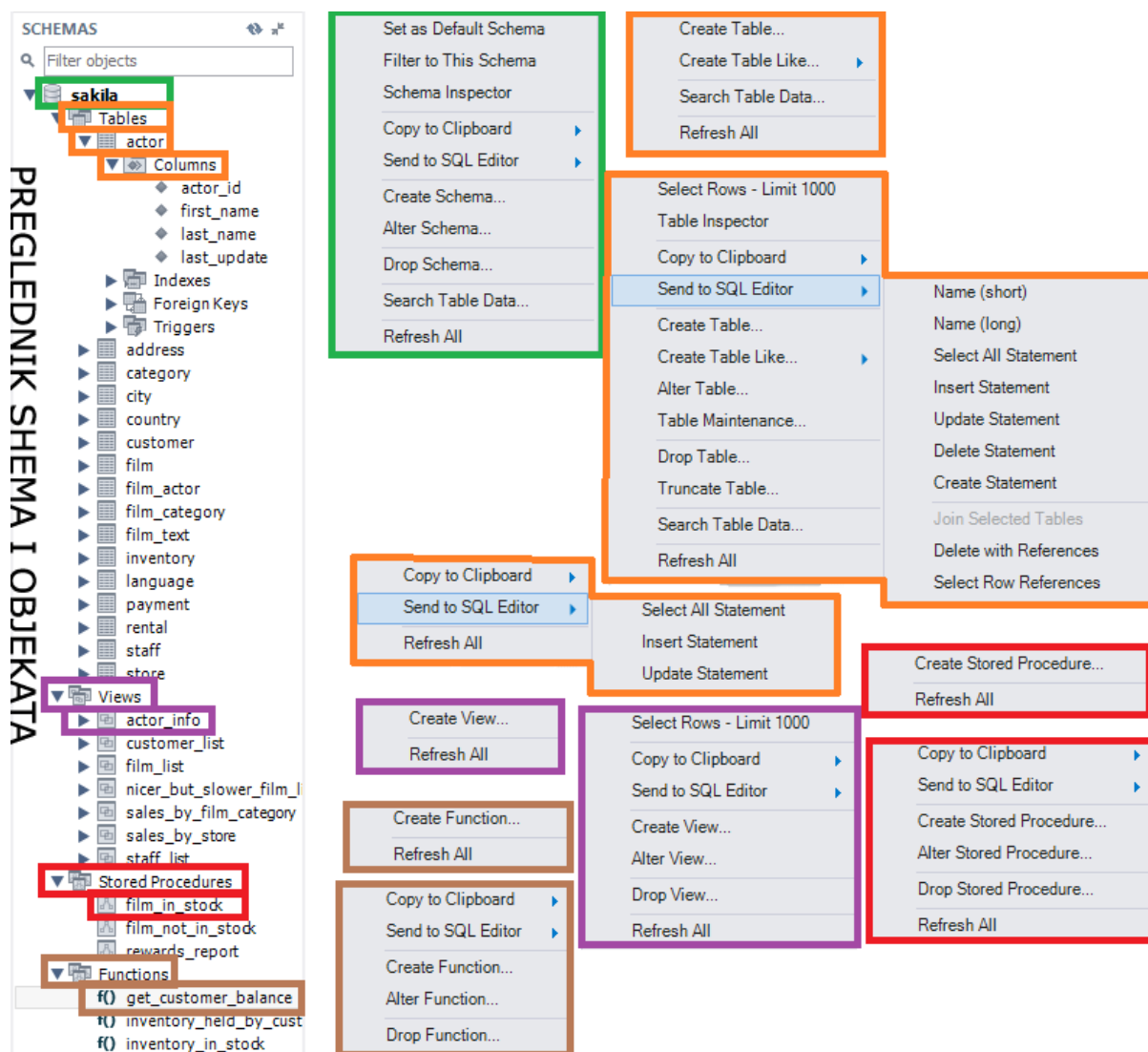


Prikaz popisa baza podataka, pripadajućih objekata te ugrađenih opcija nad objektima i shemama

Svaka od prikazanih baza podataka, kao i pripadajući objekti sadrže skup sebi specifičnih funkcija i opcija kojima je moguće pristupiti posredstvom pomoćnih ikona (vidi prethodnu sliku: A – Schema Inspector, B – Table Inspector, C, D). Komponenta A na slici predstavlja alat za detaljan pregled svojstava sheme baze dok komponenta B ima istu ulogu primijenjenu na tablice u bazi. Komponenta C ima ulogu prethodno opisane komponente za uređivanje definicije tablica u bazi podataka.

Dodatan izvor opcija za svaku pojedinu komponentu u okviru preglednika shema i objekata MySQL poslužitelja otvara se desnim klikom tipke miša na pojedini objekt, odnosno bazu. Tako se za odabranu shemu u proizašlom izborniku nalaze funkcije za kreiranje, brisanje i izmjenu sheme (Create

Schema, Drop Schema, Alter Schema), postavljanje defaultne sheme trenutne MySQL konekcije (Default Schema) i dr. Isto vrijedi i za ostale objekte odabrane baze podataka, poput tablica, pogleda, spremljenih procedura te funkcija, gdje svaki od njih sadrži vlastiti skup funkcija pod okupom odgovarajućih izbornika. Na primjer, tablice kao objekti sadrže opcije za kreiranje novih tablica (Create Table, Create Table Like), pretraživanje sadržaja tablica, osvježavanje prikaza sadržaja i dr. Svaki od objekata raščlanjen je na svoje sastavne dijelove koji također imaju odgovarajuće funkcije što je prikazano sljedećom slikom.

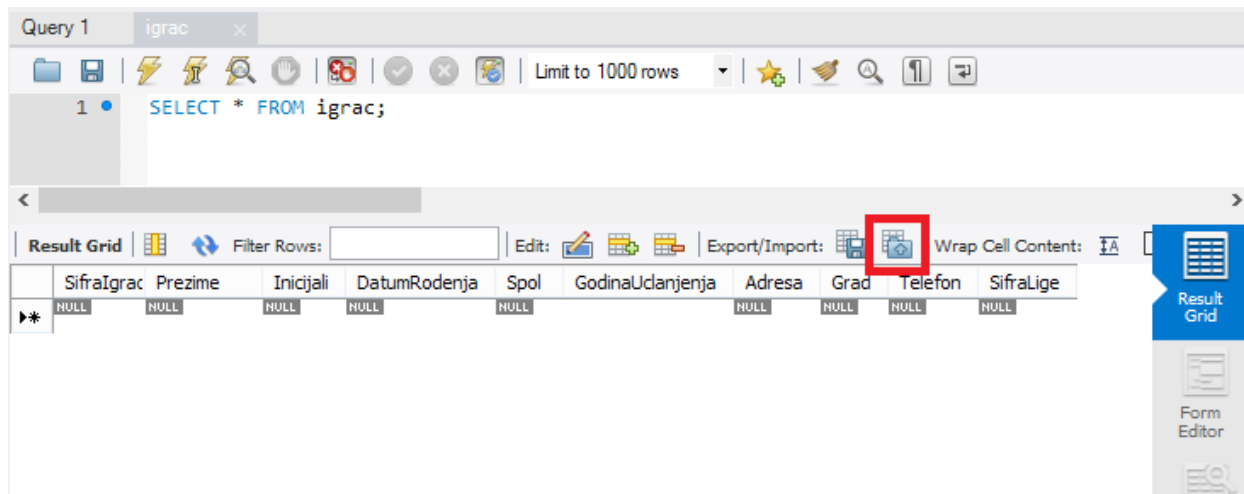


Preglednik shema i objekata sa specifičnim opcijama i funkcijama pojedinih komponenta

#### 14.4.4 Uvoz podataka iz CSV datoteka

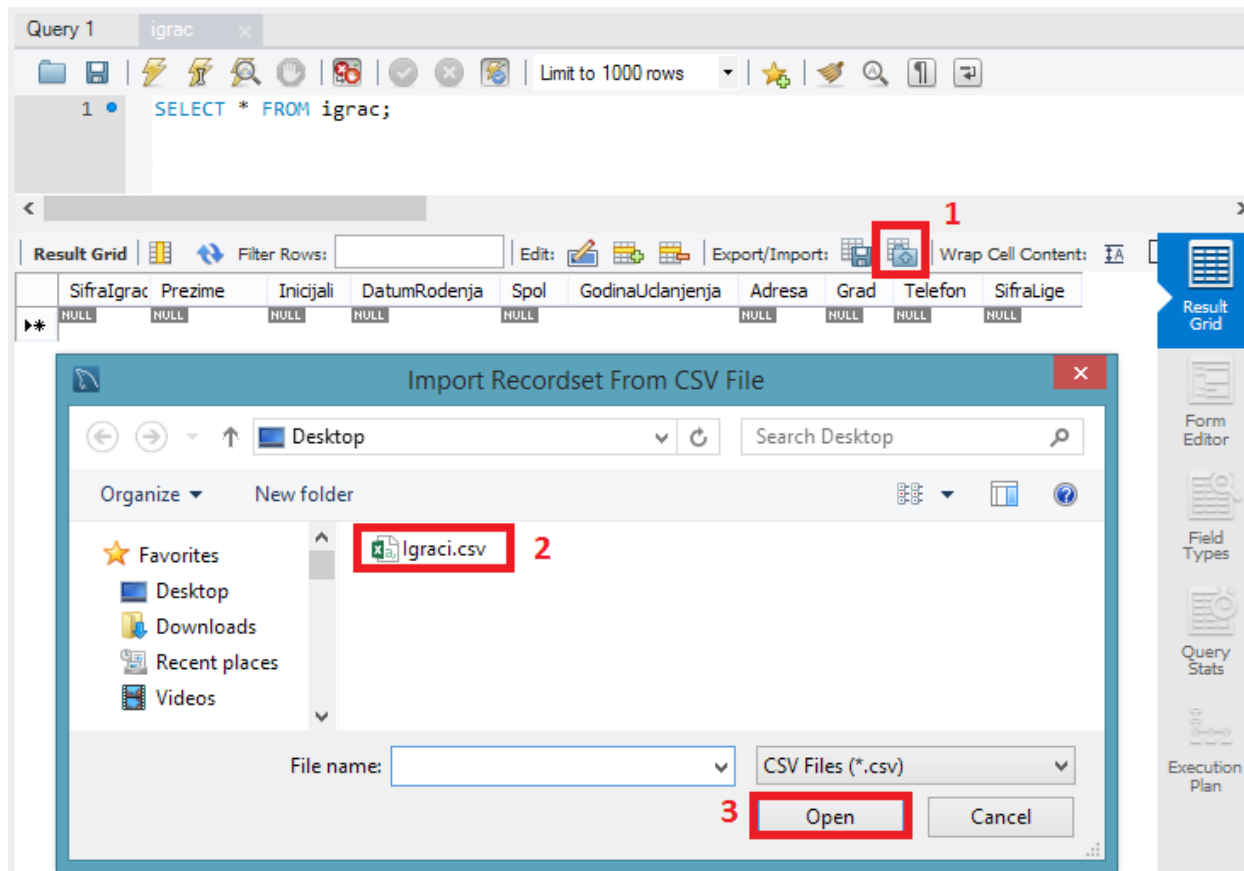
MySQL Workbench sadrži alat za uvoz podataka u tablice iz vanjskih CSV datoteka. U slučaju potreba ove vježbe podaci će se uvoziti iz CSV datoteka nastalih izvozom i obradom podataka MS Access baze podataka poznate iz vježbe 11. Ovaj alat zamjenjuje upis i izvršenje niza INSERT INTO komandi jednostavnim odabirom odgovarajućih CSV datoteka, čime je postupak ubacivanja entiteta značajno skraćen, a mogućnost pogrešaka svedena na minimum. Najprije je u Query Editoru potrebno selektirati

željenu tablicu jednostavnim upitom tipa `SELECT * FROM naziv_tablice;` čime se otvara prozor prikaza sadržaja rezultata upita (Result Grid).



Prikaz sadržaja tablice Igrac

U alatnoj traci prozora s prikazom rezultata upita, nalazi se funkcija za uvoz podataka iz vanjskih datoteka (označeno na prethodnoj slici). Pokretanjem navedene funkcije potrebno je pronaći željenu datoteku te potvrditi dodavanje klikom na dugme Open.



Odabir željene CSV datoteke

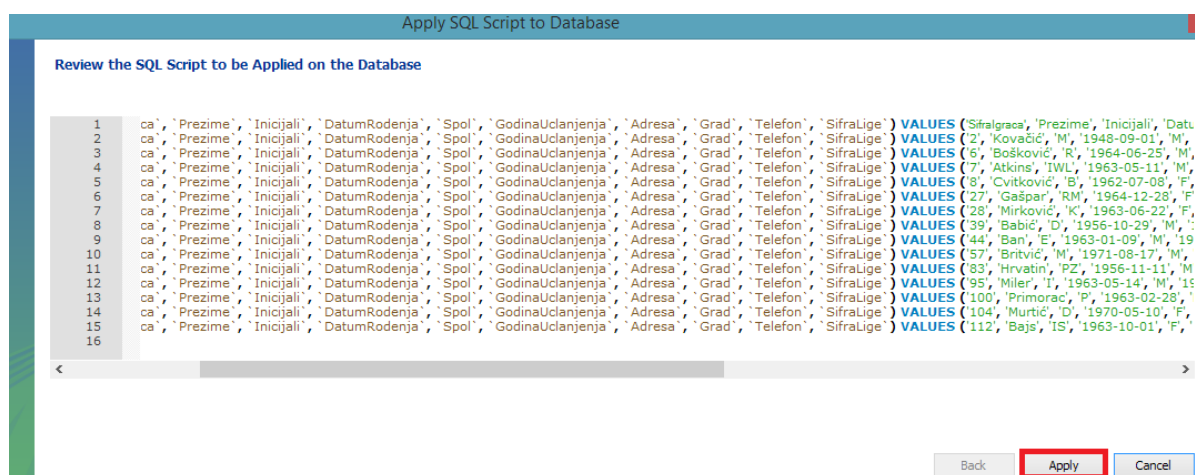


Nakon učitavanja sadržaja odabrane datoteke, potrebno je izbaciti prvi redak (entorku) jer sadrži podatke nazivlja zaglavlja tablice iz koje je obavljen izvoz podataka, a to svakako nisu podaci koji odgovaraju stvarnim vrijednostima jedne entorke. Postupak izbacivanja navedene entorke iz tablice opisan je sljedećom slikom. Prema slici, prvo je potrebno označiti željnu entorku, a potom ju obrisati klikom na ikonu označenu brojem 2.

	SifraIgrac	Prezime	Inicijali	DatumRodjenja	Spol	GodinaUclanjenja	Adresa	Grad	Telefon	SifraLige
1	SifraIgrac...	Prezime	Inicijali	DatumRodjenja	Spol	GodinaUclanjenja	Adresa	Grad	Telefon	SifraLige
2		Kovačić	M	1948-09-01	M	1975	Vinkov...	Vuk...	033-2...	2411
6		Bošković	R	1964-06-25	M	1977	Mažur...	Vuk...	033-4...	8467
7		Atkins	IWL	1963-05-11	M	1981	Stank...	Vuk...	033-3...	
8		Cvitković	B	1962-07-08	F	1980	Matije...	Križ...	033-4...	2983
27		Gašpar	RM	1964-12-28	F	1983	Vukov...	Viro...	043-2...	2513
28		Mirković	K	1963-06-22	F	1983	Tri pile	Osijek	031-6...	
39		Babić	D	1956-10-29	M	1980	Meštr...	Vuk...	033-3...	
44		Ban	E	1963-01-09	M	1980	Sveto...	Križ...	033-3...	1124
57		Britvić	M	1971-08-17	M	1985	Andrij...	Vuk...	033-4...	6409
83		Hrvatini	PZ	1956-11-11	M	1982	Bauer...	Vuk...	033-3...	1608
95		Miler	I	1963-05-14	M	1972	Dravska	Pito...	033-8...	
100		Primorac	P	1963-02-28	M	1979	Vočar...	Vuk...	033-4...	6524
104		Murtić	D	1970-05-10	F	1984	Prera...	Viro...	043-9...	7060
112		Bajs	IS	1963-10-01	F	1984	Boško...	Dar...	031-5...	1319
▶*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Izbacivanje prvog retka iz prikazane tablice

Osim izbacivanja entorke, moguće je i uređivati vrijednosti atributa ostalih entorki, sve dok se postupak konačno ne potvrdi dugmetom Apply. Nakon toga se postupak uvoza podataka u tablicu MySQL baze podataka smatra uspješno obavljen.




Potvrda procesa ubacivanja entorki iz vanjske CSV datoteke

## 14.5 Sadržaj i tijek izvođenja vježbe

U ovoj vježbi je potrebno kreirati bazu podataka u MySQL-u, na temelju primjera iz 11. vježbe, te na osnovu toga pristupiti rješavanju sljedećih zadataka. Prethodno je potrebno provesti instalaciju navedenog SUBP-a prema prikazanim uputama vježbe. Budući se ova vježba teorijski oslanja na sve prethodne vježbe, za uspješno rješavanje potrebno je prethodno proučiti teoriju, odnosno upute dosadašnjih vježbi.

### 14.5.1 Tijek izvođenja vježbe

1. Preuzmite instalacijski paket MySQL Server Community Edition sa lokacije službenog distributora: <http://dev.mysql.com/downloads/mysql/>
2. Provedite cjelokupan postupak instalacije te u postupak uključite instalaciju MySQL Workbench vizualnog alata za upravljanje MySQL RDBMS-om.
3. Kreirajte novu MySQL konekciju te novu bazu podataka i pripadajuće objekte i ograničenja (entitetska i referencijalna) na temelju baze podataka iz vježbe 11.
4. Kreirajte bazu podataka bpvj14 sljedećom komandom: `create database bpvj14;`
5. Označite bazu podataka kao trenutno defaultnu, dvostrukim klikom na naziv bpvj14 u Pregledniku shema i objekata u lijevom dijelu prozora (prije toga osvježite stanje prikaza klikom na ikonu Refresh smještenu pokraj naziva sekcije SCHEMAS.



```

Query 1 x
Limit to 1000 rows
1 create table igrac(
2   SifraIgraca int,
3   Prezime text,
4   Inicijali text,
5   DatumRodjenja date,
6   Spol text,
7   GodinaUclanjenja int,
8   Adresa text,
9   Grad text,
10  Telefon text,
11  SifraLige text,
12  constraint pk1 primary key(SifraIgraca)
13 );
14
15 create table kazna(
16   SifraKazne integer primary key,
17   SifraIgraca integer,
18   DatumPlacanja date,
19   Iznos decimal(5,0),
20   constraint fk1 foreign key(SifraIgraca) references igrac(SifraIgraca)
21 );
22
23 create table TeniskiOdbor(
24   SifraIgraca int,
25   PocetakMandata date,
26   KrajMandata date,
27   Funkcija text,
28   constraint pk2 primary key(SifraIgraca,PocetakMandata),
29   constraint fk2 foreign key(SifraIgraca) references igrac(SifraIgraca)
30 );
31
32 create table Tim(
33   SifraTima int primary key,
34   SifraIgraca int,
35   Divizija text,
36   constraint fk3 foreign key(SifraIgraca) references igrac(SifraIgraca)
37 );
38
39 create table Utakmica(
40   SifraUtakmice int primary key,
41   SifraTima int,
42   SifraIgraca int,
43   DobiveniSetovi int,
44   IzgubljeniSetovi int,
45   constraint fk4 foreign key(SifraTima) references Tim(SifraTima),
46   constraint fk5 foreign key(SifraIgraca) references igrac(SifraIgraca)
47 );

```

- Uvezite podatke iz priloženih CSV datoteka u upravo kreiranu bazu podataka.
- Kreirajte upit kojim ćete prikazati prosječan ukupan iznos kazni za sve igrače iz Vukovara i Križevaca.

Expr1000
85,00 kn

- Kreirajte upit kojim ćete prikazati šifru igrača, prezime i ukupan broj kazni te ukupan broj timova kojima je pojedini igrač bio kapetan i to za one igrače koji su zaradili barem jednu kaznu te imaju status kapetana.

SifraIgraca	Prezime	BrojKazni	BrojTimova
6	Bošković	1	1
27	Gašpar	2	1

- Kreirajte upit kojim ćete prikazati šifru igrača i ukupan broj kazni za svakog igrača koji je odigrao barem jedan meč.

Sifralgraca	BrojKazni
2	
6	1
8	1
27	2
44	3
57	
83	
104	1
112	

10. Kreirajte upit kojim ćete prikazati sumu svih kazni za sljedeće grupe vremenskih perioda:

- Grupa 1: od 1/1/1980 do 30/6/1982
- Grupa 2: od 1/7/1981 do 31/12/1982
- Grupa 3: od 1/1/1983 do 31/12/1984

Rezultat upita prikazan je sljedećom slikom:

GRUPA	SUMA
1	225,00 kn
2	30,00 kn
3	225,00 kn

11. Kreirajte upit kojim ćete prikazati iznos kazne te sumu te kazne i svih kazni manjeg iznosa (kumulativna suma).

SifraKazne	Iznos	Expr1002
1	100,00 kn	100,00 kn
2	75,00 kn	175,00 kn
3	100,00 kn	275,00 kn
4	50,00 kn	325,00 kn
5	25,00 kn	350,00 kn
6	25,00 kn	375,00 kn
7	30,00 kn	405,00 kn
8	75,00 kn	480,00 kn

12. Kreirajte upit kojim ćete prikazati za svaku kaznu njezinu šifru, iznos te postotni udio u ukupnom iznosu kazni.

SifraKazne	Iznos	Expr1002
1	100,00 kn	20,83333333333333
2	75,00 kn	15,625
3	100,00 kn	20,83333333333333
4	50,00 kn	10,41666666666667
5	25,00 kn	5,208333333333333
6	25,00 kn	5,208333333333333
7	30,00 kn	6,25
8	75,00 kn	15,625

13. Kreirajte upit kojim ćete prikazati prosječan broj igrača koji žive u gradu.

Expr1000
2,33333333

14. Kreirajte upit kojim ćete prikazati šifru tima, pripadajuću diviziju te ukupan broj igrača koji su igrali za taj tim.

SifraTima	Divizija	BrojIgraca
1	prva	8
2	druga	5

15. Kreirajte upit kojim ćete prikazati šifru igrača, prezime i razliku između godine učlanjenja u klub i prosječne godine učlanjenja u klub.

SifraIgraca	Prezime	Expr1002
2	Kovačić	-5,35714285714289
6	Bošković	-3,35714285714289
7	Atkins	0,64285714285711
8	Cvitković	-0,35714285714289
27	Gašpar	2,64285714285711
28	Mirković	2,64285714285711
39	Babić	-0,35714285714289
44	Ban	-0,35714285714289
57	Britvić	4,64285714285711
83	Hrvatini	1,64285714285711
95	Miler	-8,35714285714289
100	Primorac	-1,35714285714289
104	Murtić	3,64285714285711
112	Bajs	3,64285714285711

16. Kreirajte upit kojim ćete prikazati šifru igrača, prezime te razliku godine učlanjenja u klub i prosječne godine učlanjenja u klub za sve igrače iz istog grada.

SifraIgraca	Prezime	Expr1002
2	Kovačić	-4,85714285714289
6	Bošković	-2,85714285714289
7	Atkins	1,14285714285711
8	Cvitković	0
27	Gašpar	-0,5
28	Mirković	0
39	Babić	0,14285714285711
44	Ban	0
57	Britvić	5,14285714285711
83	Hrvatini	2,14285714285711
95	Miler	0
100	Primorac	-0,85714285714289
104	Murtić	0,5
112	Bajs	0

## 14.6 Završna pitanja i zadaci

### 14.6.1 Zadaci za samostalno produblivanje znanja

- Nakon vježbe diskutirajte sa kolegama oko rješenja vježbe. Diskutirajte na temelju argumenata baziranih na uvodu u vježbu te na temelju predavanja iz kolegija baze podataka.

## 14.7 Literatura i dodatni materijali

1. <https://www.mysql.com/>
2. <http://www.mysqltutorial.org/>
3. <https://dev.mysql.com/doc/refman/5.6/en/tutorial.html>
4. <https://www.mysql.com/products/workbench/>

