Usporedba performansi Redis i MSSQL baze podataka na primjeru poslovne aplikacije

Ćuić, Filip

Undergraduate thesis / Završni rad

2018

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:245900

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-04-17**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni preddiplomski studij

USPOREDBA PERFORMANSI REDIS I MSSQL BAZE PODATAKA NA PRIMJERU POSLOVNE APLIKACIJE

Završni rad

Filip Ćuić

Osijek, 2018.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

Obrazac Z1P - Obrazac za ocjenu završnog rada na preddiplomskom sveučilišnom studiju

Osijek, 04.09.2018.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada

Ime i prezime studenta:	Filip Ćuić
Studij, smjer:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R3760, 21.09.2017.
OIB studenta:	49678680047
Mentor:	Doc.dr.sc. Ivica Lukić
Sumentor:	
Sumentor iz tvrtke:	
Naslov završnog rada:	Usporedba performansi Redis i MSSQL baze podataka na primjeru poslovne aplikacije
Znanstvena grana rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Predložena ocjena završnog rada:	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 3 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene mentora:	04.09.2018.
Datum potvrde ocjene Odbora:	

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 07.09.2018.

Ime i prezime studenta:	Filip Ćuić
Studij:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R3760, 21.09.2017.
Ephorus podudaranje [%]:	3%

Ovom izjavom izjavljujem da je rad pod nazivom**: Usporedba performansi Redis i MSSQL baze podataka na primjeru poslovne aplikacije**

izrađen pod vodstvom mentora Doc.dr.sc. Ivica Lukić

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

IZJAVA

Ja, Filip Ćuić, OIB: 49678680047, student/ica na studiju: Preddiplomski sveučilišni studij Računarstvo, dajem suglasnost Fakultetu elektrotehnike, računarstva i informacijskih tehnologija Osijek da pohrani i javno objavi moj **završni rad**:

Usporedba performansi Redis i MSSQL baze podataka na primjeru poslovne aplikacije

u javno dostupnom fakultetskom, sveučilišnom i nacionalnom repozitoriju.

Osijek, 07.09.2018.

_____

potpis

Sadržaj rada

1. UVOD 1
1.1. Zadatak završnog rada 2
2. ALATI 3
2.1. Microsoft Visio 2016 3
2.2. Microsoft SQL Server Management Studio 17 4
2.2.1. SQL 5
2.3. Mockaroo proizvođač stvarnih podataka 7
2.4. Redis 8
2.4.1. Redis stringovi 9
2.4.2. Redis liste 10
2.4.3. Redis skupovi 10
2.4.4. Redis poredani skupovi 11
2.4.5. Redis hashevi 12
2.5. Microsoft Visual Studio 2017 12
2.5.1. Programski jezik C# 13
3. IZRADA ZAVRŠNOG RADA 15
3.1. Izrada relacijskog modela podataka 15
3.1.2. Zadani problem 15
3.1.2. Utvrđivanje i analiza normaliziranog modela podataka (ER modela) „Fitness sustava“ 16
3.1.3. Pretvaranje ER modela u relacijski model podataka 19
3.2. Izrada relacijskog modela podataka u programu Microsoft SQL i mjerenje performansi programa 27
3.2.1. Upiti za definiranje baze podataka Fitness sustava 27
3.2.2. Upiti za upravljanje bazom podataka Fitness sustava 28
3.2.3. Mjerenje performansi MSSQL baze podataka 29
3.3. Unos u Redis nerelacijsku bazu podataka i mjerenje performansi baze 32
3.3.1. Opis rada s Redis serverom preko C# konzolne aplikacije 32
3.3.2. Opis mjerenja performansi 34
3.3.3. Mjerenje performansi Redis baze podataka 35
3.4. Usporedba MSSQL-a i Redisa 37
3.5. Interpretacija dobivenih rezultata 38

4. ZAKLJUČAK 40
LITERATURA 41
SAŽETAK 43
ABSTRACT 44
ŽIVOTOPIS 45
PRILOZI 46

1. UVOD

Zadatak završnog rada je usporediti performanse relacijske (MSSQL) i nerelacijske baze podataka (Redis) na primjeru poslovne aplikacije, odnosno baze podataka fitness sustava. Glavni cilj rada je ustanoviti je li pogodnije raditi poslovnu aplikaciju prema relacijskim modelima i koristiti relacijsku bazu podataka, ili bez relacijskog modela preko korištenja nerelacijske baze podataka. Performanse koje će se ispitivati su : brzina unosa, pristupa i brisanja podataka, te preglednost, sigurnost i struktura baze podataka.

Pri izradi relacijske baze podataka (MSSQL) fitness sustava nužno je poznavanje SQL jezika, teorijsko poznavanje modeliranja podataka, odnosno poznavanje modela entitet-veza, te objektnih modela podataka. Sva ta znanja stečena su na kolegiju „Baze podataka“. Za izradu nerelacijske baze podataka (Redis) koristilo se znanje programiranja u programskom jeziku C# stečeno na kolegiju „Objektno-orijentirano programiranje“. Izrada završnog rada dijeli se na 3 dijela :

1) Definiranje ER (*eng. Entity Relationship*) modela problema, te izrada ER i relacijskog modela podataka u programu „Microsoft Visio“ radi lakšeg unosa u relacijsku bazu podataka (MSSQL)
2) Izrada relacijskog modela podataka (tablica) u programu Microsoft SQL Server Management Studio, unošenje različitog broja n-torki u tablice pomoću Mockaroo proizvođača stvarnih podataka (*eng. Mockaroo Realistic Data Generator*), te mjerenje performansi baze i obavljenih operacija
3) Unos podataka u Redis nerelacijsku bazu podataka preko programa Microsoft Visual Studio u programskom jeziku C#, te mjerenje performansi baze i obavljenih operacija preko alata Redis Benchmark (mjerit će se i simulirane operacije).

U drugom poglavlju objasnit će se alati koji su se koristili pri izradi rada, a u trećem poglavlju detaljan postupak izrade rada, te će se interpretirati i usporediti dobiveni rezultati. U četvrtom poglavlju će se donijeti zaključak.

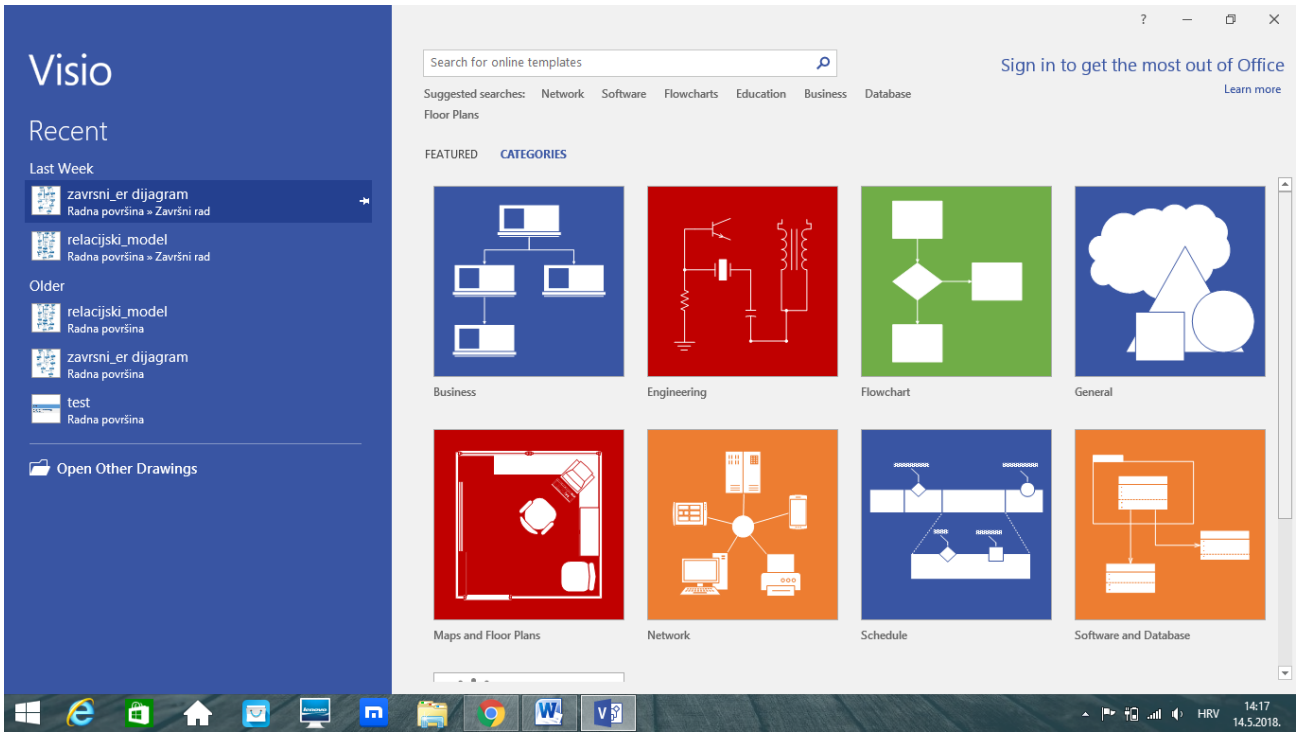
1.1. Zadatak završnog rada

Zadatak završnog rada je napraviti bazu podataka fitness sustava koji pomaže svojim klijentima vođenje i pregled podataka koji se tiču njihovog zdravlja i pravilne tjelovježbe. Također, aplikacija služi službenicima teretane u vođenju podataka o njihovim klijentima, te trenerima za pomoć klijentima u ostvarenju njihovih ciljeva. U radu će se također opisati osnovne značajke korištenih alata (Microsoft Visio, Mockaroo, Microsoft Visual Studio) i osnovne značajke baza podataka koje se koriste (MSSQL, Redis).

2. ALATI

2.1. Microsoft Visio 2016

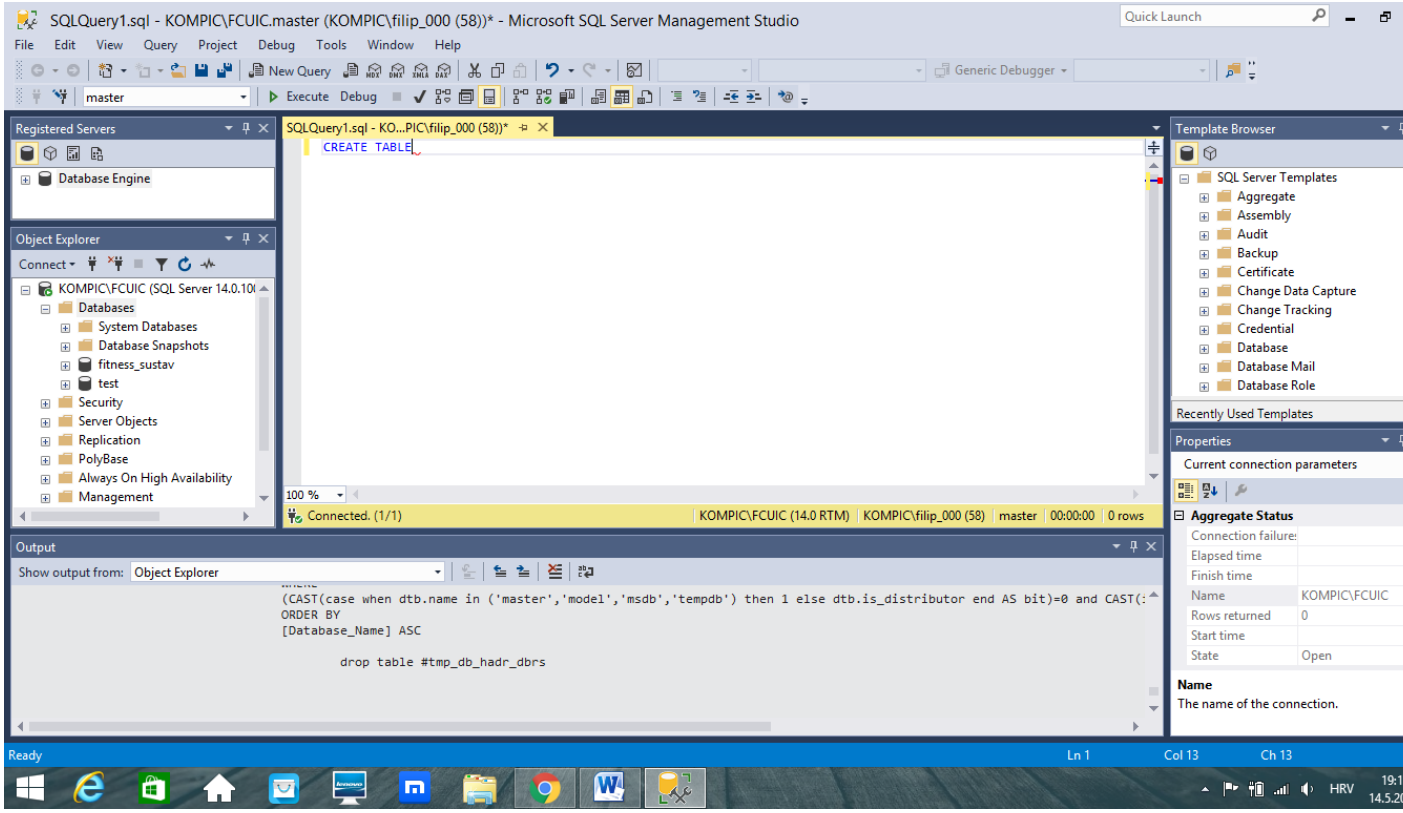
Microsoft Visio je programsko okruženje namijenjeno za razvoj dijagrama i 2D grafičkih elemenata. Prvu verziju je razvila tvrtka Shapeware Corporation 1992. godine, a od 2000. godine dio je Microsoft Office programskog paketa. Najnovija verzija Visio programske podrške izdana je 2016. godine i sadrži preko tisuću vrsta predložaka za korisničke dijagrame. Predlošci za dijagrame podijeljeni su u osam osnovnih kategorija među kojima su dijagrami za prikaz : poslovanja, inženjerskih shemi, dijagrama toka, karata i tlocrta, mreža, rasporeda, te programske podrške i baze podataka. Iz navedenog se može zaključiti da programsko okruženje ima široku primjenu u poslovanju i svim granama inženjerstva.



Slika 2.1. Prikaz početnog sučelja programskog okruženja Microsoft Visio 2016.

2.2. Microsoft SQL Server Management Studio 17

Microsoft SQL Server Management Studio 17 je integrirana okolina za pristup, konfiguraciju, upravljanje, razvoj i administraciju SQL servera i razvoj njegovih objekata. Da bi korisnik pristupio početnom sučelju SQL Servera, mora proći kroz ovjeru autentičnosti, čime pokazuje da je ovlašten za korištenje usluga servera. Jedan dio početnog sučelja je pretraživač objekata (*eng. Object Explorer*) preko kojeg korisnik ima pregled svih definiranih baza podataka (*eng. Databases*), sigurnosnih stavki (*eng. Security*), objekata servera (*eng. Server Objects*) i mnogih drugih stavki servera. Drugi dio početnog sučelja je prostor za upite preko kojega se unosi SQL kod. Izvršavanjem tog koda se definira i upravlja relacijskim modelom podataka. Treba napomenuti da definiranje i upravljanje podacima nije nužno kroz izvršavanje SQL upita, tu mogućnost također nude opcije u pretraživaču objekata. U narednom potpoglavlju bit će objašnjene osnove SQL jezika kojim će se obavljati navedene radnje.



Slika 2.2. Prikaz početnog sučelja Microsoft SQL Server Management Studia nakon autentikacije korisnika

2.2.1. SQL

SQL (*eng. Structured Query Language*) je jezik za :

1) Kreiranje, ažuriranje i upite nad bazom podataka
2) Dodavanje, brisanje i mijenjanje redaka (atributa) definirane tablice i n-torki (objekata) sheme
3) Kontrolu pristupa bazi podataka i objektima sheme

Njihovim korištenjem dobiva se relacijska baza podataka čija je glavna shema u obliku tablice, odnosno svaki definirani objekt sa svojim vrijednostima je jedna instanca definirane tablice. Tablica (relacija) se sastoji od svojeg zaglavlja (skup atributa) i od vrijednosti pojedinog atributa. Stupci relacije sa svojim vrijednostima predstavljaju skup svih atributa tablice, a svaki redak tablice predstavlja jednu n-torku. Primjerice, tablica (relacija) čovjek sastoji se od zaglavlja koje obuhvaća atribute ime, prezime, starost itd. Sve vrijednosti jednog retka tablice čovjek (Primjerice, ime: Marko, prezime: Markić, godine : 50) predstavljat će jednog čovjeka, odnosno n-torku tablice.

Upiti SQL-a dijele se na :

1) DDL (*eng. Data Definition Language*) – upiti za upravljanje tablicama (Primjer : CREATE/ALTER/DROP/TRUNCATE TABLE)
2) DML (*eng. Data Manipulation Language*) – upiti za dohvaćanje, unos, uklanjanje, ažuriranje i upravljanje objektima definiranim pomoću DDL-a (Primjer : SELECT, INSERT, UPDATE, DELETE, BULK INSERT, MERGE itd.)
3) DCL (*eng. Data Control Language*) – upiti za kontrolu pristupa podacima pohranjenim u bazu podataka i omogućavanje sigurnosti podataka (Primjer : GRANT, REVOKE, EXECUTE AS statement/clause)
4) TCL (*eng. Transaction Control Language*) – omogućava grupiranje upita u logičke transakcije (Primjer : BEGIN/COMMIT/ROLLBACK TRANSACTION)

Osnovni tipovi podataka u SQL-u :

1) Tekstualni tipovi podataka : CHAR, VARCHAR, TINYTEXT, TEXT, BLOB, MEDIUMTEXT, MEDIUMBLOB, LONGTEXT, LONGBLOB, ENUM('a','b','c'), SET

2) Brojčani tipovi podataka : TINYINT, SMALLINT, MEDIUMINT, INT, BIGINT, FLOAT, DOUBLE, DECIMAL
3) Tipovi podataka za bilježenje nadnevaka i vremena : DATE, DATETIME, TIMESTAMP, TIME, YEAR

Ostali tipovi podataka : SQL_VARIANT, UNIQUEIDENTIFIER, XML, CURSOR

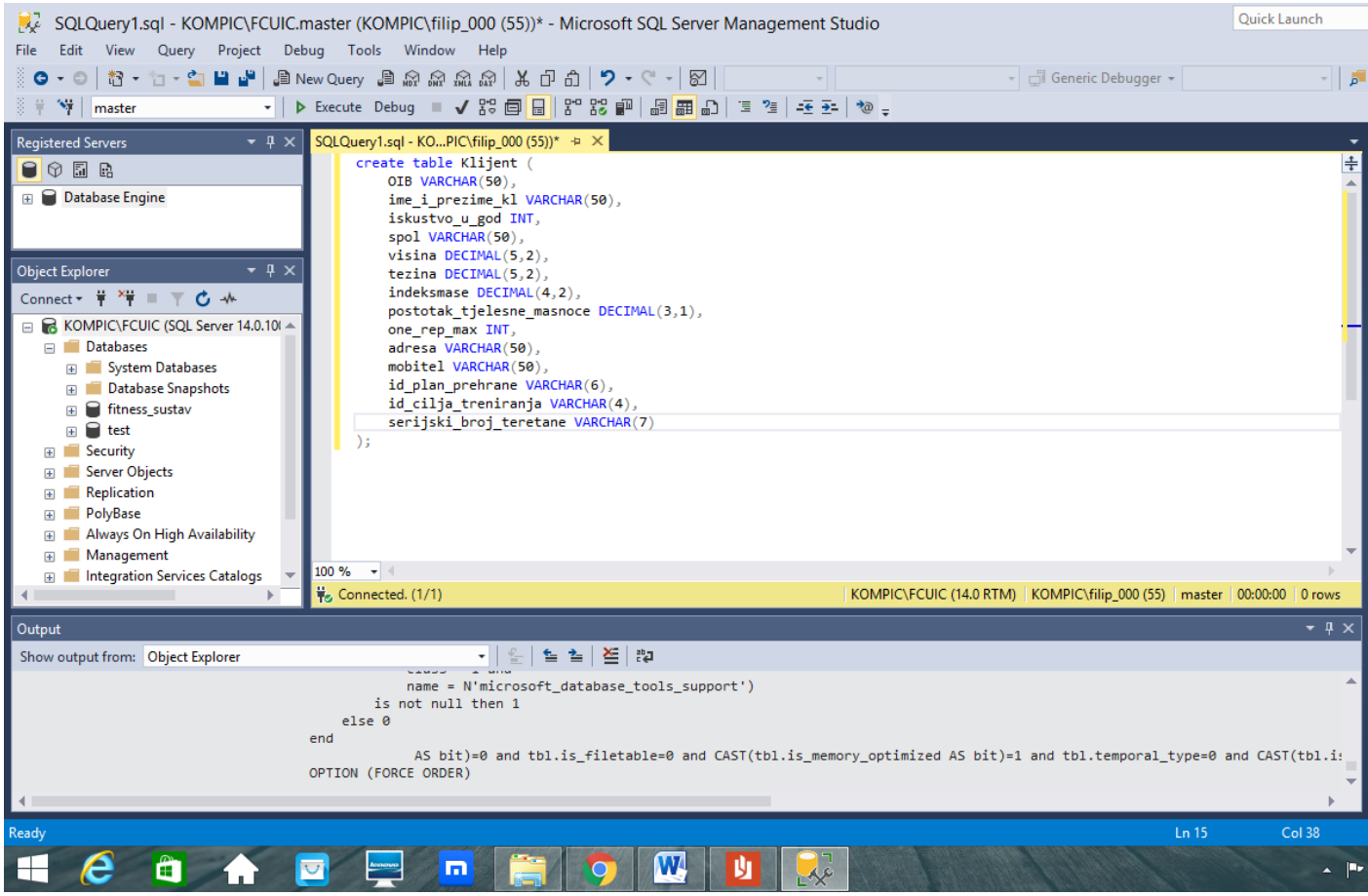
Domena je skup mogućih vrijednosti koje su sadržane unutar određenog atributa ili polja. Tip podatka sam definira raspon vrijednosti koje atribut može sadržavati, no ako se ta domena želi dodatno specificirati i ograničiti, koriste se **Ograničenja (*eng. Constraints*)** :

1) PRIMARY KEY – primarni ključ tablice, atribut koji određuje jedinstvenost objekta unutar tablice, odnosno jednoznačno određuje svaku n-torku unutar tablice
2) UNIQUE – jedinstveni ključ tablice, atribut koji je jedinstven za svaku n-torku tablice
3) FOREIGN KEY – strani ključ tablice, određuje atribut koji upućuje (referencira) na primarni ključ neke druge tablice čime se uspostavlja veza između n-torki (objekata) različitih tablica (relacija)
4) CHECK – služi za ograničenje raspona vrijednosti atributa tablice, moguće je nabrajanje vrijednosti (Primjer : CHECK spol IN('M','F')) i ograničavanje pomoću logičkih operatora jednako (=),manje od (<), manje ili jednako od (<=), veće od (>), veće ili jednako od (>=) (Primjer : CHECK (godine_ucenika>=6 AND godine_ucenika<15)

Ugrađene funkcije u SQL-u djeluju nad određenim vrijednostima i daju drugu vrijednost, preslikavaju jedan ili više elemenata iz domene atributa u neku drugu domenu.

Funkcije SQL-a dijele se u dvije skupine :

1) Skalarne funkcije – djeluju nad jednom vrijednošću i kao rezultat daju jednu vrijednost (Primjerice, funkcija SQRT(broj) kao rezultat daje drugi korijen zadanog broja)
2) Agregatne funkcije – djeluju nad skupom vrijednosti određenog atributa tablice i kao rezultat daju jednu vrijednost, mogu se koristiti samo unutar SELECT i HAVING upita (Primjer : SELECT AVG(ocjena) FROM student WHERE (ocjena>1) ; Upit će dati prosječnu vrijednost jednog atributa(ocjene) svih n-torki kojima je vrijednost atributa ocjena veća od 1)

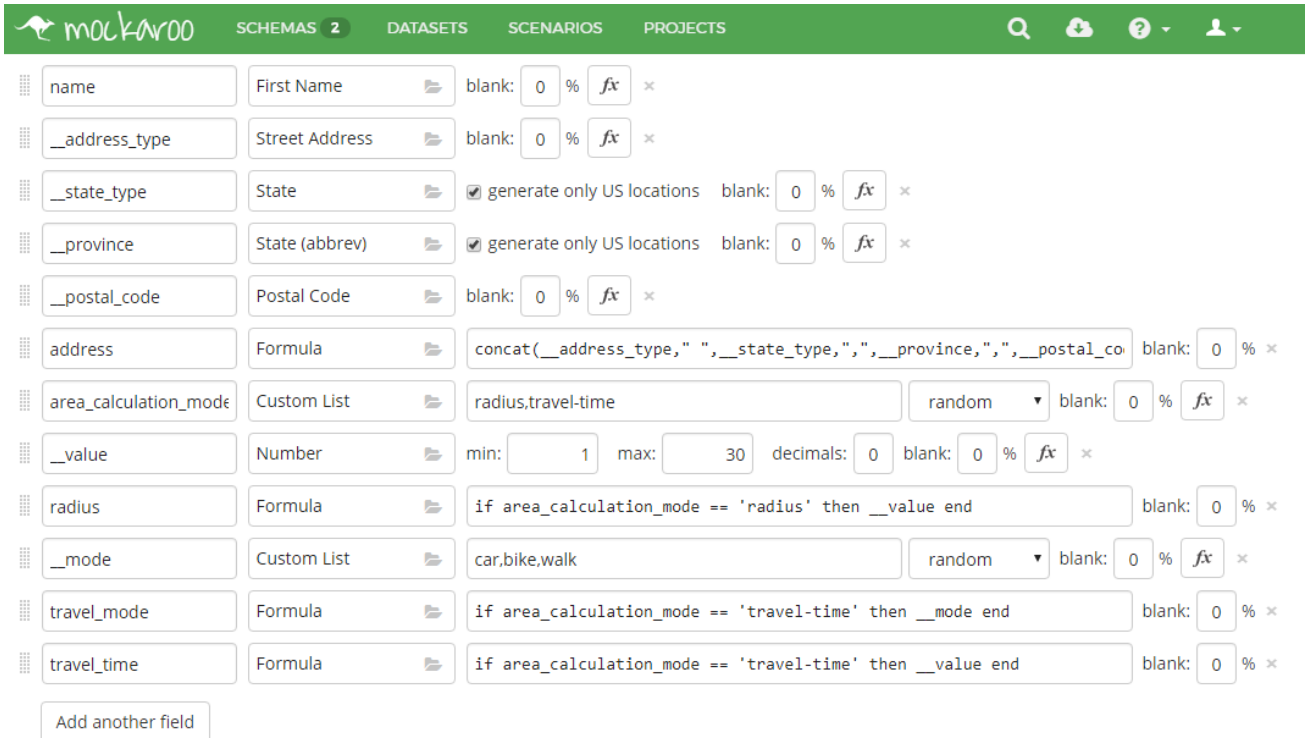


Slika 2.3. Prikaz stvaranja tablice u SQL jeziku

2.3. Mockaroo proizvođač stvarnih podataka

Mockaroo proizvođač stvarnih podataka (*eng. Mockaroo Realistic data generator*) je „open-source“ proizvođač nasumičnih podataka dostupan preko internet preglednika. Razvio ga je Mark Brocato za potrebe testiranja svoje baze podataka o kliničkim ispitivanjima i laboratorijskim eksperimentima. Zbog nedostatka obrazaca upotrebe nad bazom podataka odlučio je napraviti aplikaciju koja će svojim korisnicima omogućiti stvaranje i popunjavanje baze stvarnim podacima. Aplikacija dostupna preko internetskog preglednika počela je dobivati sve više korisnika koji su slali zahtjeve o stvaranju novih tipova podataka. Korisnici mogu specificirati tipove podataka za pojedine atribute, kao i raspon stvorenih nasumičnih podataka. Aplikacija sadrži 142 različita tipa podataka. Korisnik je u mogućnosti stvoriti upite koji u njihove baze unose ogromne količine podataka samo jednim klikom miša. Moguće je odrediti tipove stvorenih upita (SQL, Cassandra CQL, XML itd.), te broj stvorenih n-torki relacije (najviše 1000 bez plaćenog korištenja). Stvaranjem korisničkog računa korisnik dobiva

mogućnost stvaranja više tablica (relacija) odjednom i povezivanje tih tablica stranim ključevima. Prije preuzimanja stvorenog koda, korisnik može odrediti želi li stvoriti podatke za unos i tablicu u koju se podaci unose ili samo podatke za unos u već stvorenu tablicu. Za stvaranje više vlastitih tablica mora se registrirati na stranicu i prijaviti za korištenje usluga Mockaroo-a. Korištenjem besplatne verzije alata može se stvoriti najviše 1000 podataka (n-torki) za svaku definiranu relaciju.



Slika 2.4. Prikaz sučelja Mockaroo-a za registrirane korisnike

2.4. Redis

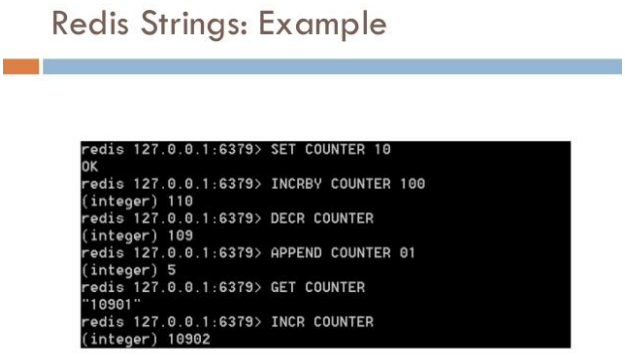
Redis (skraćeno od REmote DIctionary Server) je jako brza nerelacijska (NoSQL) baza podataka koja pohranjuje ključne vrijednosti (*eng.keys*) u pet mogućih struktura podataka:

1) Stringovi
2) Liste (*eng. lists*)
3) Skupovi (*eng.sets*)
4) Poredani skupovi (*eng.sorted sets*)
5) Hashevi

Razvio ga je Salvatore Sanfillipo 2009. godine. Omogućuje pohranu podataka tako što se vrijednosti (*eng.value*) spremaju unutar pripadajućih ključeva (*eng.key*). Baza podataka je otvorenog koda (*eng.open-source*) što znači da je pristup preuzimanju i korištenju baze potpuno otvoren i besplatan. Glavne odlike Redisa su brzina unosa i dohvaćanja podataka, atomske operacije nad podacima (sigurno i brzo mijenjanje unesenih podataka), te velik broj programskih jezika koje podržava (C, C++, Java, C# itd.). Neki od nedostataka Redisa su nemogućnost upita nad vrijednostima objekata (ne može se pristupiti n-torki gdje je, primjerice, vrijednost jednaka danom broju), te primitivan način komuniciranja s Redis serverom (komunikacija linija-po-linija), zbog čega se na Redis spaja preko raznih drugih programskih jezika i platformi u cilju slanja brojnih upita prema serveru odjednom. Napisan je u ANSI C programskom jeziku i može se koristiti na gotovo svim operacijskim sustavima (najpogodniji za Linux). Koriste ga najpoznatije internet stranice (Twitter, Instagram, StackOverflow, Github itd.) za spremanje podataka o sesijama korisnika na web-stranicama i za upravljanje predmemorijom web-stranica.

2.4.1. Redis stringovi

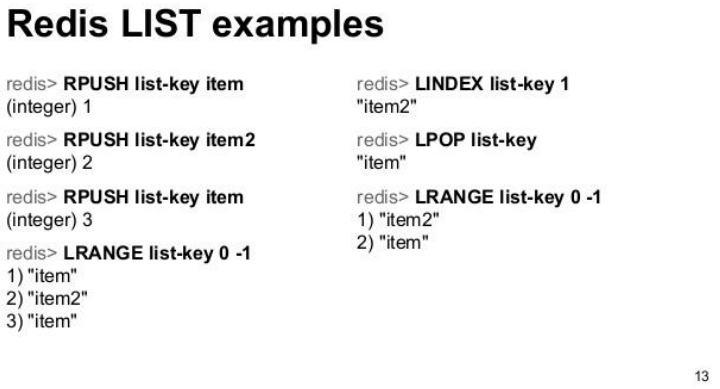
Stringovi su najjednostavnije dostupne strukture podatka u Redisu. Rade na principu ključ-vrijednost (*eng.key-value storage*), odnosno navođenjem ključa u njega se sprema vrijednost ili dohvaća spremljena odgovarajuća vrijednost. Moguće su razne druge operacije nad njima kao što su dohvaćanje dijela spremljene vrijednosti, mijenjanje vrijednosti, postavljanje vremena istjecanja ključa i sl. Važno je napomenuti da vrijednosti spremljene u Redis stringove nisu nužno polja znakova (*eng.strings*) nego i cjelobrojne i decimalne brojčane vrijednosti.



Slika 2.5. Prikaz nekih od operacija nad redis stringovima

2.4.2. Redis liste

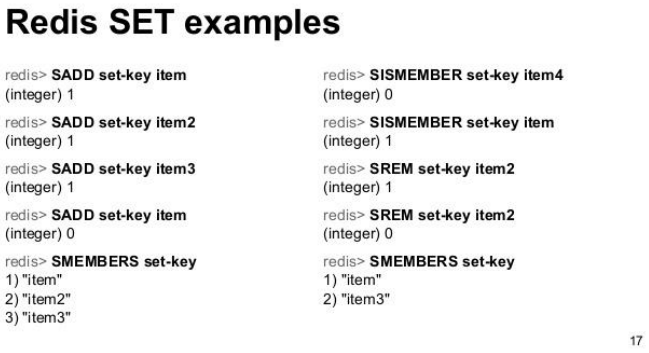
Redis liste omogućuju pohranjivanje i rukovanje poljem vrijednosti koje je spremljeno u određenom ključu. One su zapravo liste stringova poredane po redoslijedu unosa. Liste zadržavaju slijed unosa i elementima se može pristupiti preko njihovih indeksa. Uklanjanje elemenata liste naredbom LTRIM procedura je koja ima O(n) složenost, gdje je n broj vrijednosti odgovarajuće liste koje se žele izbrisati. Elementi liste mogu se dodavati slijeva (naredba LPUSH) i zdesna (naredba RPUSH). Liste mogu sadržavati do 4 milijuna elemenata. Idealne su za bilježenje aktivnosti korisnika aplikacija.



Slika 2.6. Prikaz naredbi za unos elemenata u listu, dohvaćanje svih elemenata iz liste, dohvaćanje elementa liste na danom indeksu, te brisanje prvog elementa liste slijeva

2.4.3. Redis skupovi

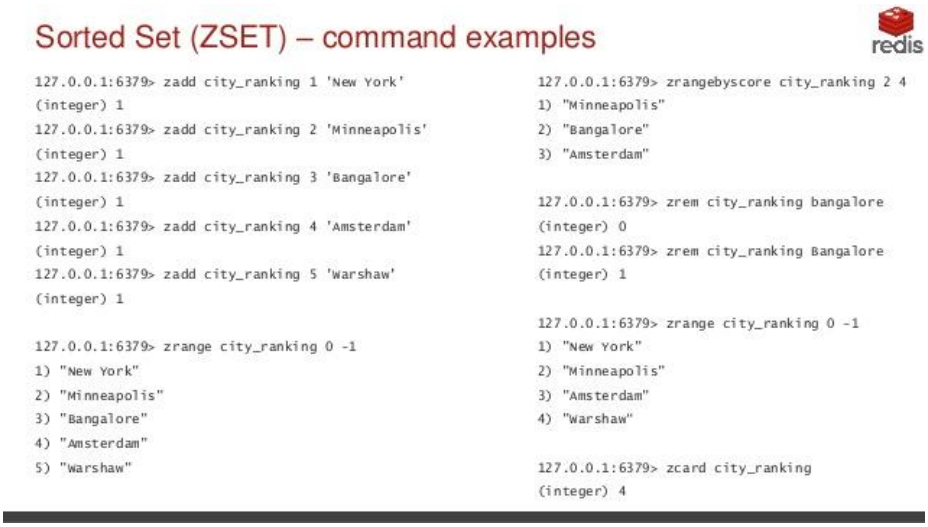
Skupovi (*eng.Sets*) se koriste za pohranu jedinstvenih vrijednosti. Nazivaju se skupovima jer se nad više skupova mogu obavljati operacije kao nad matematičkim skupovima (unije, presjeci i razlike). Za procedure dodavanja, brisanja i dohvaćanja podataka iz skupa složenost je O(1), odnosno procedure se odvijaju jednako brzo neovisno o broju podataka unutar skupa. Idealni su za, primjerice, označavanje i praćenje podataka o vremenu pristupa korisnika društvenih mreža gdje ponavljanja vrijednosti podataka nema.



Slika 2.7. Naredbe za dodavanje, dohvaćanje, ispitivanje postojanosti i brisanje podataka iz skupova

2.4.4. Redis poredani skupovi

Poredani skupovi (*eng. Sorted sets*) su skupovi u kojima je svakom elementu unutar skupa dodijeljen određeni brojčani iznos (*eng. score*) na temelju čega se vrši sortiranje i svrstavanje elemenata unutar skupa. Poredak elemenata u skupu radi se od elementa s najmanjim iznosom do elementa s najvećim iznosom, tj. poredak je uzlazan. Ova struktura podataka idealna je za modeliranje tablica poredaka.



Slika 2.8. Primjer ubacivanja elemenata s dodijeljenim brojčanim vrijednostima u poredani skup, te dohvaćanje elemenata poredanog skupa

2.4.5. Redis hashevi

Redis hashevi su struktura podataka kojom se mogu predstavljati n-torke (objekti). Mogu se promatrati kao objekti koji sadržavaju varijable različitih vrijednosti. N-torka se definira na način : **HMSET kljuc polje1 vrijednost1 polje2 vrijednost2**. Imena svih varijabli i njihovih vrijednosti unutar hasha dobivaju se naredbom **HGETALL kljuc**, a same vrijednosti unutar hasha naredbom **HVALS kljuc**. Prilikom izrade rada koristit će se isključivo ova struktura podataka za prikaz n-torki u Redis bazi podataka.

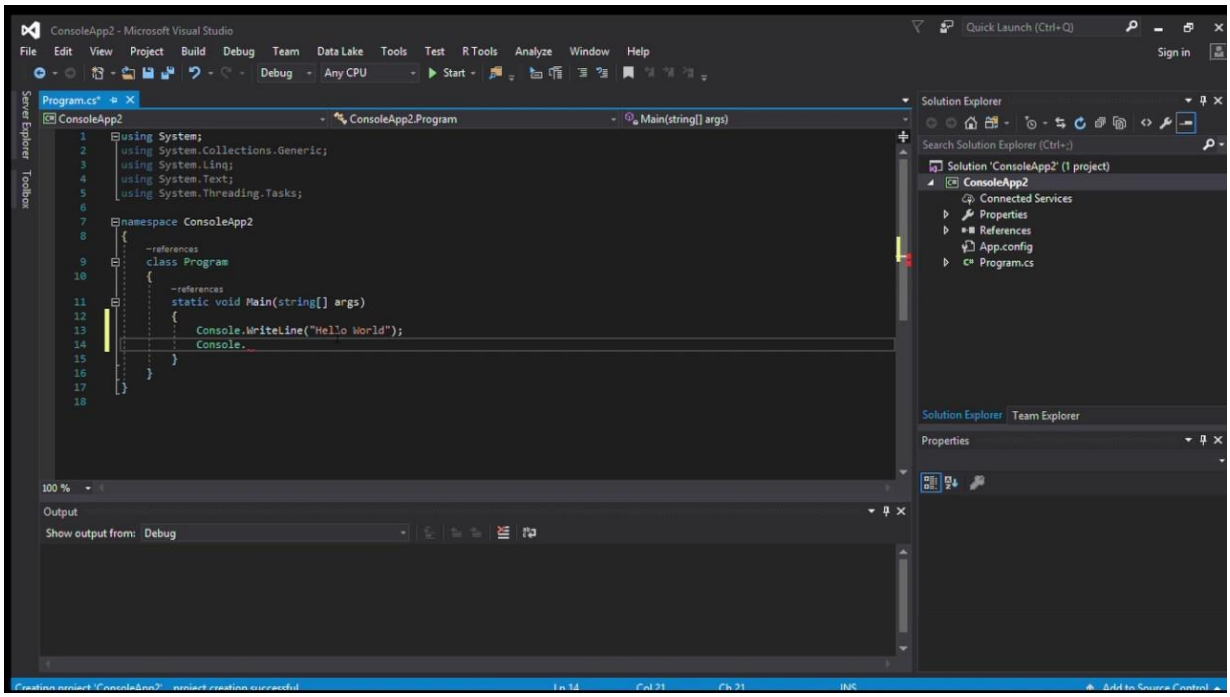
```
127.0.0.1:6379> HMSET COVJEK IME "PETAR" PREZIME "PERIC" OIB "11111111111" GODINE 28

127.0.0.1:6379> HGETALL COVJEK

1) "IME 2) "PETAR" 3) "PREZIME" 4) "PERIC" 5) "OIB" 6) "11111111111" 7) "GODINE" 8) "28"

127.0.0.1:6379> HVALS COVJEK

1) "PETAR" 2) "PERIC" 3) "11111111111" 4) "28"

127.0.0.1:6379> HKEYS COVJEK

1) "IME" 2) "PREZIME" 3) "OIB" 4) "GODINE"

127.0.0.1:6379> HGET COVJEK IME

"PETAR"
```

Slika 2.9. Primjer definiranja hasha, dohvaćanja svih polja i vrijednosti hasha, te dohvaćanje vrijednosti željenog polja preko Redis naredbenog retka (*eng. Redis command line interface*)

2.5. Microsoft Visual Studio 2017

Visual Studio je ugrađeno razvojno okruženje (*eng. Integrated Development Environment*) koje je razvila tvrtka Microsoft. Služi za razvoj računalnih programa, web-stranica, web-aplikacija i mobilnih aplikacija. Prva verzija razvijena je 1997. godine, sadržavala je programski jezik J++ i razvojno okruženje Visual InterDev koje je sadržavalo alate za izgradnju web-stranica, pronalaženje pogrešaka (*eng.debugger*) i upravljanje bazom podataka. Napisan je u programskim jezicima C i C++. Najnovije verzije podržavaju 36 programskih jezika, te sadrže ugrađeni nalaznik pogrešaka (*eng. debugger*) koji radi na razini izvornog i strojnog koda. Najjednostavnija verzija programa Community Edition Visual Studio dostupna je svakome za

preuzimanje i upotrebu. Osim sadržavanja velikog broja programskih jezika (C, C++, C#, JavaScript...) okruženje omogućuje i preuzimanje raznih funkcijskih biblioteka što olakšava pisanje programa korištenjem gotovih funkcija. Zbog navedenih razloga, Visual Studio je bez konkurencije najkorištenije okruženje razvojnih programera. Zadnja verzija okruženja izdana je 1. Lipnja 2018. godine.



Slika 2.10. Prikaz sučelja Microsoft Visual Studia

2.5.1. Programski jezik C#

Programski jezik C# (C Sharp) je objektno-orijentirani programski jezik opće namjene kojeg je razvila grupa razvojnih programera tvrtke Microsoft na čelu s Andersom Hejlsbergom. Sintaksa jezika slična je jezicima C, C++ i Javi (krajevi izjava završavaju s točka-zarezom, grupiranje izjava je pomoću kovrčastih zagrada, varijable se deklariraju znakom '=' i uspoređuju s '=='). Sadrži sve značajke objektno-orijentiranih jezika kao što su polimorfizam i nasljeđivanje, klase i objekti, imenici itd. Od tipova podataka mogući su logički tip (bool), znakovni tipovi (char, string), cjelobrojni tipovi s predznakom (sbyte, int, short, long), cjelobrojni tipovi bez predznaka (ushort, uint, ulong), brojčani tipovi jednostruke/dvostruke preciznosti (float, double), te decimalni tip (decimal). Operatori jezika C# dijele se na aritmetičke, logičke i relacijske. Kreiranje polja, matrica i objekata ostvaruje se naredbom new, navođenjem tipa podatka i veličine unutar uglatih zagrada (Primjerice : int[] polje = new int[20]). Ispis podatka spremljenog

u varijabli vrši se pomoću naredbe Console.WriteLine (Primjer : Console.WriteLine ("vrijednost1:{0},vrijednost2:{1}", var1,var2);), a upis podatka u varijablu preko standardnog ulaza pomoću naredbe Console.ReadLine() (Primjer : int a = Console.ReadLine();). Visual Studio omogućuje korištenje C# programskog jezika u kreiranju aplikacije Windows Formi (*eng. Windows Forms Application*) i u kreiranju konzolne aplikacije (*eng. Console Application*) u .NET okruženju.

3. IZRADA ZAVRŠNOG RADA

3.1. Izrada relacijskog modela podataka

Za sljedeći opis problema „Fitness sustava“ potrebno je napraviti normalizirani model podataka s pripadajućim entitetima, vezama između postojećih entiteta i atributima entiteta.

3.1.2. Zadani problem

Fitness sustav sastoji se od klijenta. Svaki klijent prati jedan plan prehrane i ima jedan cilj treniranja (gubljenje težine, dobivanje težine, poboljšanje u određenom sportu itd.). Svaki plan prehrane sastoji se od više vrsta hrane, a jedan plan prehrane može pratiti više klijenata. Svaki klijent može odrađivati više tipova treninga, a svaki tip treninga može obavljati više klijenata. Svaki tip treninga sastoji se od više vježbi. Obavljanjem svake vježbe izgrađuje se jedan ili više mišića, a svaki mišić može se izgraditi obavljanjem više različitih vježbi. Za obavljanje određene vježbe se ne mora koristiti niti jedna vrsta opreme i može se koristiti više vrsta opreme, a svaka vrsta opreme može se koristiti za obavljanje više različitih vježbi. Teretana se unutar fitness sustava sastoji od više vrsta opreme, više trenera i više klijenata. Svaki trener može trenirati više klijenata, kao što svakog klijenta može trenirati jedan ili više trenera, ili niti jedan trener. O svakom klijentu vode se podaci o njegovom/njezinom OIB-u, imenu i prezimenu, iskustvu vježbanja u godinama, spolu, visini, težini, indeksu mase, postotku tjelesne masnoće, one rep maxu (koliku težinu najviše može podići obavljajući bilo koju vježbu), adresi i mobitelu. Za svaki plan prehrane vode se podaci o jedinstvenom identifikatoru plana, broju kalorija, trajanju u mjesecima, te ukupnoj količini bjelančevina, masti i ugljikohidrata. Za hranu se vode podaci o jedinstvenom identifikatoru hrane, kalorijskoj vrijednosti, količini bjelančevina, masti i ugljikohidrata, te o komentaru stručnjaka o kvaliteti vrste hrane. Za ciljeve treniranja se vode podaci o jedinstvenom identifikatoru cilja, imenu cilja, prosječnom trajanju koje je potrebno za njegovo ostvarivanje u mjesecima, te kratki opis cilja. Za entitet trener vode se podaci o jedinstvenom identifikatoru, imenu i prezimenu, trenerskom iskustvu u godinama, adresi, mobitelu, e-mailu i specijalnosti. Za teretanu se vode podaci o serijskom broju teretane

(jedinstveni identifikator svake teretane), lokaciji, datumu otvorenja, broju članova i trenera, te o iznosu mjesečne članarine. Svaka vrsta opreme treba se sastojati od podataka o identifikatora opreme, naziva opreme, te postavljene težine. Vježba sadrži atribute identifikator vježbe, ime vježbe, broj serija i ponavljanja, te opis izvođenja vježbe. Svaki Tip treninga sastoji se od identifikatora tipa treninga, imena tipa treninga, broja serija i ponavljanja tijekom određenog tipa treninga, trajanja u minutama, datuma izvođenja i utrošenih kalorija. Svaki mišić u bazi treba imati podatke o identifikatoru mišića, nazivu mišića, latinskom nazivu mišića, tipu mišićnog vlakna (Tip I ili II) i o lokaciji u tijelu.

3.1.2. Utvrđivanje i analiza normaliziranog modela podataka (ER modela) „Fitness sustava“

Entiteti : KLIJENT, PLAN_PREHRANE, HRANA, CILJ_TRENIRANJA, TRENER, TERETANA, TIP_TRENINGA, VJEŽBA, MIŠIĆ, VRSTA_OPREME

Atributi pojedinog entiteta (uz podcrtane primarne ključeve entiteta):

KLIJENT (OIB, IME_I_PREZIME_KL, ISKUSTVO_U_GOD, SPOL, VISINA, TEZINA, INDEKSMASE, POSTOTAK_TJELESNE_MASNOCE, ONE_REP_MAX, ADRESA, MOBITEL)

PLAN_PREHRANE (ID_PLANA, BROJ_KALORIJA, TRAJANJE_U_MJESECIMA, KOLICINA_BJELANČEVINA,KOLICINA_MASTI,KOLICINA_UGLJIKOHIDRATA)

HRANA (IDENTIFIKATOR, KALORIJSKA_VRIJEDNOST, KOLICINA_BJELANCEVINA, KOLICINA_MASTI, KOLICINA_UGLJIKOHIDRATA, KOMENTAR_STRUCNJAKA)

CILJ_TRENIRANJA (ID_CILJA, IME, PROSJECNO_TRAJANJE_U_MJESECIMA, KRATKI_OPIS)

TRENER (ID_TRENERA, IME_I_PREZ_TRENERA, ISKUSTVO_U_GOD, ADRESA, MOBITEL, E-MAIL, SPECIJALNOST)

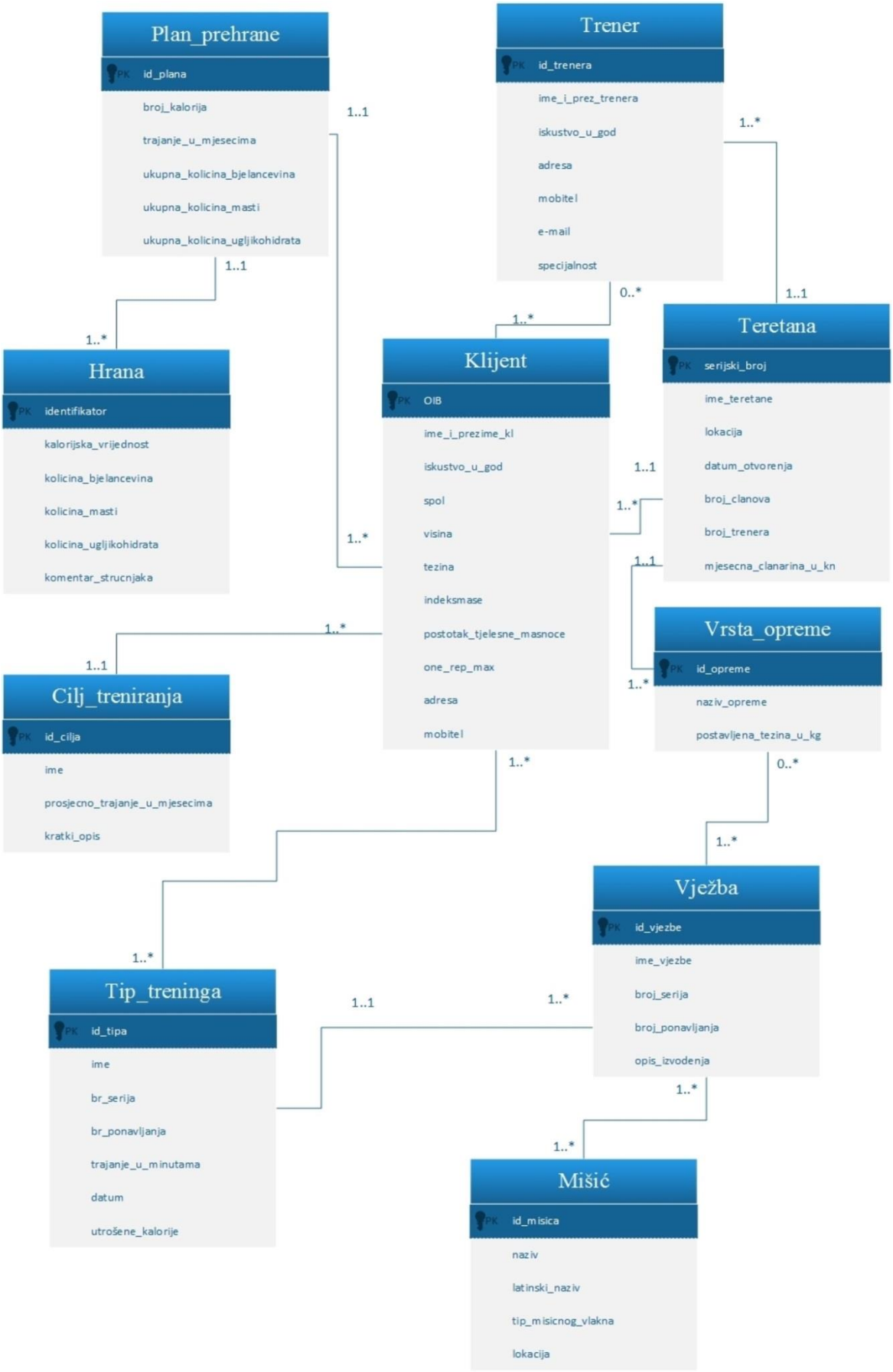
TERETANA (SERIJSKI_BROJ, LOKACIJA, DATUM_OTVORENJA, BROJ_CLANOVA, BROJ_TRENERA, MJESECNA_CLANARINA)

VRSTA_OPREME (ID_OPREME, NAZIV_OPREME, POSTAVLJENA_TEZINA)

VJEŽBA (ID_VJEZBE, IME_VJEZBE, BROJ_SERIJA, BROJ_PONAVLJANJA, OPIS_IZVODENJA)

TIP_TRENINGA (ID_TIPA, IME, BR_SERIJA, BR_PONAVLJANJA, TRAJANJE_U_MINUTAMA, DATUM, UTROŠENE_KALORIJE)

MIŠIĆ (ID_MISICA, NAZIV, LATINSKI_NAZIV, TIP_MISICNOG_VLAKNA, LOKACIJA)



Slika 3.1. Prikaz Entity-Relationship modela podataka Fitness sustava napravljenog u programu Microsoft Visio

3.1.3. Pretvaranje ER modela u relacijski model podataka

Za potrebe unosa u relacijsku (SQL) bazu podataka potrebno je pretvoriti normalizirani ER (*eng. Entity-Relationship*) model u relacijski model u kojem su entiteti međusobno povezani preko svojih stranih ključeva i/ili preko entiteta-veze. Prema teoriji modeliranja podataka, entiteti koji su spojeni vezom jedan-prema-više u ER modelu bit će spojeni tako da će se primarni ključ entiteta sa strane veze jedan dodati kao strani ključ u entitet sa strane više. Nadalje, entiteti koji su u vezi više-prema-više u ER modelu u relacijskom će biti spojeni preko novog entiteta-veze koji će sadržavati primarne ključeve obaju rubnih entiteta koji će zajedno činiti primarni ključ entiteta-veze.

Prikaz Entiteta i entiteta-veza zajedno s njihovim atributima (primarni ključevi entiteta podcrtani su ravnom, a strani valovitom linijom) :

Entiteti :

KLIJENT (OIB, IME_I_PREZIME_KL, ISKUSTVO_U_GOD, SPOL, VISINA, TEZINA, INDEKSMASE, POSTOTAK_TJELESNE_MASNOCE, ONE_REP_MAX, ADRESA, MOBITEL, ID_PLAN_PREHRANE, ID_CILJA_TRENIRANJA, SERIJSKI_BROJ_TERETANE)

PLAN_PREHRANE (ID_PLANA, BROJ_KALORIJA, TRAJANJE_U_MJESECIMA, KOLICINA_BJELANČEVINA,KOLICINA_MASTI,KOLICINA_UGLJIKOHIDRATA)

HRANA (IDENTIFIKATOR, KALORIJSKA_VRIJEDNOST, KOLICINA_BJELANCEVINA ,KOLICINA_MASTI, KOLICINA_UGLJIKOHIDRATA, KOMENTAR_STRUCNJAKA, ID_PLANA_PREHRANE)

CILJ_TRENIRANJA (ID_CILJA, IME, PROSJECNO_TRAJANJE_U_MJESECIMA, KRATKI_OPIS)

TRENER (ID_TRENERA, IME_I_PREZ_TRENERA, ISKUSTVO_U_GOD, ADRESA, MOBITEL, E-MAIL, SPECIJALNOST, SERIJSKI_BROJ_TERETANE_GDJE_RADI)

TERETANA (SERIJSKI_BROJ, LOKACIJA, DATUM_OTVORENJA, BROJ_CLANOVA, BROJ_TRENERA, MJESECNA_CLANARINA)

VRSTA_OPREME (ID_OPREME, NAZIV_OPREME, POSTAVLJENA_TEZINA, ID_TERETANE)

VJEŽBA (ID_VJEZBE, IME_VJEZBE, BROJ_SERIJA, BROJ_PONAVLJANJA, OPIS_IZVODENJA, ID_TIPA_TRENINGA)

TIP_TRENINGA (ID_TIPA, IME, BR_SERIJA, BR_PONAVLJANJA, TRAJANJE, DATUM, UTROŠENE_KALORIJE)

MIŠIĆ (ID_MISICA, NAZIV, LATINSKI_NAZIV, TIP_MISICNOG_VLAKNA, LOKACIJA)

Entiteti-veze (atributi su istovremeno primarni i strani ključevi podcrtani crvenom bojom):

IMA - između entiteta TRENER i KLIJENT (OIB_KLIJENTA, ID_TRENERA)

OBAVLJA - između entiteta KLIJENT i TIP TRENINGA (OIB_KLIJENTA, ID_TIPA_TRENINGA)

ZAHTJEVA - između entiteta VRSTA_OPREME i VJEŽBA (ID_OPREME, ID_VJEŽBE)

AKTIVIRA - između entiteta VJEŽBA i MIŠIĆ (ID_MISICA, ID_VJEZBE)

Tablica 3.1. Opis atributa za relacijsku bazu podataka fitness sustava (podcrtani su primarni ključevi entiteta).

Ime atributa	Tip podatka i raspon vrijednosti	Opis atributa
OIB	Niz od 11 znakova	Osobni indentifikacijski broj klijenta, jedinstveni identifikator klijenta
ime_i_prezime	Niz od 50 znakova	Ime i prezime određenog klijenta
iskustvo_u_god	Cijeli broj od 1 do 50 (uključujući dva broja)	Iskustvo vježbanja klijenta izraženo u godinama
spol	Jedan znak (M ili F)	Spol klijenta

visina	Decimalni broj od 140 do 220	Visina klijenta u centimetrima
tezina	Decimalni broj između od 40 do 200	Težina klijenta u kilogramima
indeksmase	Decimalni broj od 18.5 do 40	Klijentov indeks mase. Iskazuje je li težina klijenta prikladna njegovoj visini i dobi
postotak_tjelesne_masnoće	Decimalni broj od 5 do 50	Udio tjelesne masti u ukupnoj težini klijenta
one_rep_max	Cijeli broj od 5 do 250	Najveća kilaža koju klijent može podići jednom tijekom izvođenja bilo koje vježbe
adresa	Niz od 50 znakova	Fizička adresa klijenta
mobitel	Niz od 10 znakova	Pozivni broj klijentovog mobitela
id_plana	Niz od 6 znakova	Jedinstveni identifikator svakog plana prehrane
broj_kalorija	Decimalni broj kojeg funkcijski određuju količina bjelančevina, masti i ugljikohidrata u planu prehrane	Broj kalorija koji podrazumjeva određeni plan prehrane
trajanje_u_mjesecima	Cijeli broj od 1 do 1120	Preporučena duljina praćenja određenog plana prehrane
ukupna_kolicina_bjelancevina	Decimalni broj od 30 do 200	Količina dnevnog unosa bjelančevina u gramima koju plan prehrane nalaže
ukupna_kolicina_masti	Decimalni broj od 30 do 200	Količina dnevnog unosa masti u gramima koju plan prehrane nalaže
ukupna_kolicina_ugljikohidrata	Decimalni broj od 30 do 200	Količina dnevnog unosa ugljikohidrata u gramima koju plan prehrane nalaže
identifikator	Niz od 6 znakova	Jedinstveni identifikator svake vrste hrane

kolicina_proteina	Cijeli broj od 0 do 50	Količina proteina u gramima unutar 100 grama odgovarajuće hrane
kolicina_masti	Cijeli broj od 0 do 50	Količina masti u gramima unutar 100 grama odgovarajuće hrane
kolicina_ugljikohidrata	Cijeli broj od 0 do 50	Količina ugljikohidrata u gramima unutar 100 grama odgovarajuće hrane
kalorijska_vrijednost	Decimalni broj kojeg funkcijski određuju količina proteina, masti i ugljikohidrata	Ukupna kalorijska vrijednost hrane
komentar_strucnjaka	Niz znakova varijabilne duljine (TEXT tip podatka)	Komentar fitness stručnjaka o određenoj vrsti hrane
id_cilja	Niz od 5 znakova	Jedinstveni identifikator svakog cilja treniranja
ime	Niz od 50 znakova	Ime cilja treniranja
prosjecno_trajanje_u_mjesecima	Cijeli broj od 1 do 168	Prosječni vremenski period u mjesecima koji treba proći da bi se cilj treniranja ostvario
kratki_opis	Niz znakova varijabilne duljine (TEXT tip podatka)	Opis cilja treniranja i metodi kojima se cilj nastoji postići
id_tipa	Niz od 5 znakova	Jedinstveni identifikator svakog tipa treninga
ime	Niz od 50 znakova	Ime tipa treninga
br_serija	Cijeli broj od 1 do 40	Ukupan broj izvođenja svih vježbi prije kratke stanke
br_ponavljanja	Cijeli broj od 1 do 800	Ukupan broj izvedenih ponavljanja tijekom određenog tipa treninga
trajanje_u_minutama	Cijeli broj od 1 do 240	Ukupno trajanje jednog tipa treninga u minutama
datum	Datum od 1.1.1970. do 6.6.2018	Datum obavljanja jednog tipa treninga

utrosene_kalorije	Cijeli broj od 50 do 500	Količina utrošene energije u kalorijama tijekom obavljanja određenog tipa treninga
id_vjezbe	Niz od 5 znakova	Jedinstveni identifikator za svaku vrstu vježbe
ime_vjezbe	Niz od 50 znakova	Naziv vrste vježbe
broj_serija	Cijeli broj od 1 do 7	Broj kratkih stanki tijekom izvođenja pojedine vježbe
broj_ponavljanja	Cijeli broj od 1 do 30	Ukupan broj izvođenja pojedine vježbe
opis_izvodenja	Niz znakova varijabilne duljine (TEXT tip podatka)	Savjet i upute stručnjaka o izvođenju vježbe
id_misica	Cijeli broj od 1 do 752	Jedinstveni identifikator svakog mišića u ljudskom tijelu
naziv	Niz od 50 znakova	Naziv mišića
latinski_naziv	Niz od 50 znakova	Latinski naziv mišića
tip_misicnog_vlakna	Cijeli broj, 1 ili 2	Govori izgrađuje li se mišić bolje većim kilažama i manjim brojem ponavljanja (tip 1) ili manjim kilažama i većim brojem ponavljanja
lokacija	Niz od 50 znakova	Kazuje mjesto gdje je mišić smješten unutar ljudskog tijela
id_opreme	Niz od 5 znakova	Jedinstveni identifikator svake vrste opreme
naziv_opreme	Niz od 50 znakova	Naziv vrste opreme
postavljena_tezina_u_kg	Cijeli broj od 5 do 200	Kazuje postavljenu težinu u kilogramima na određenoj opremi (spravi)

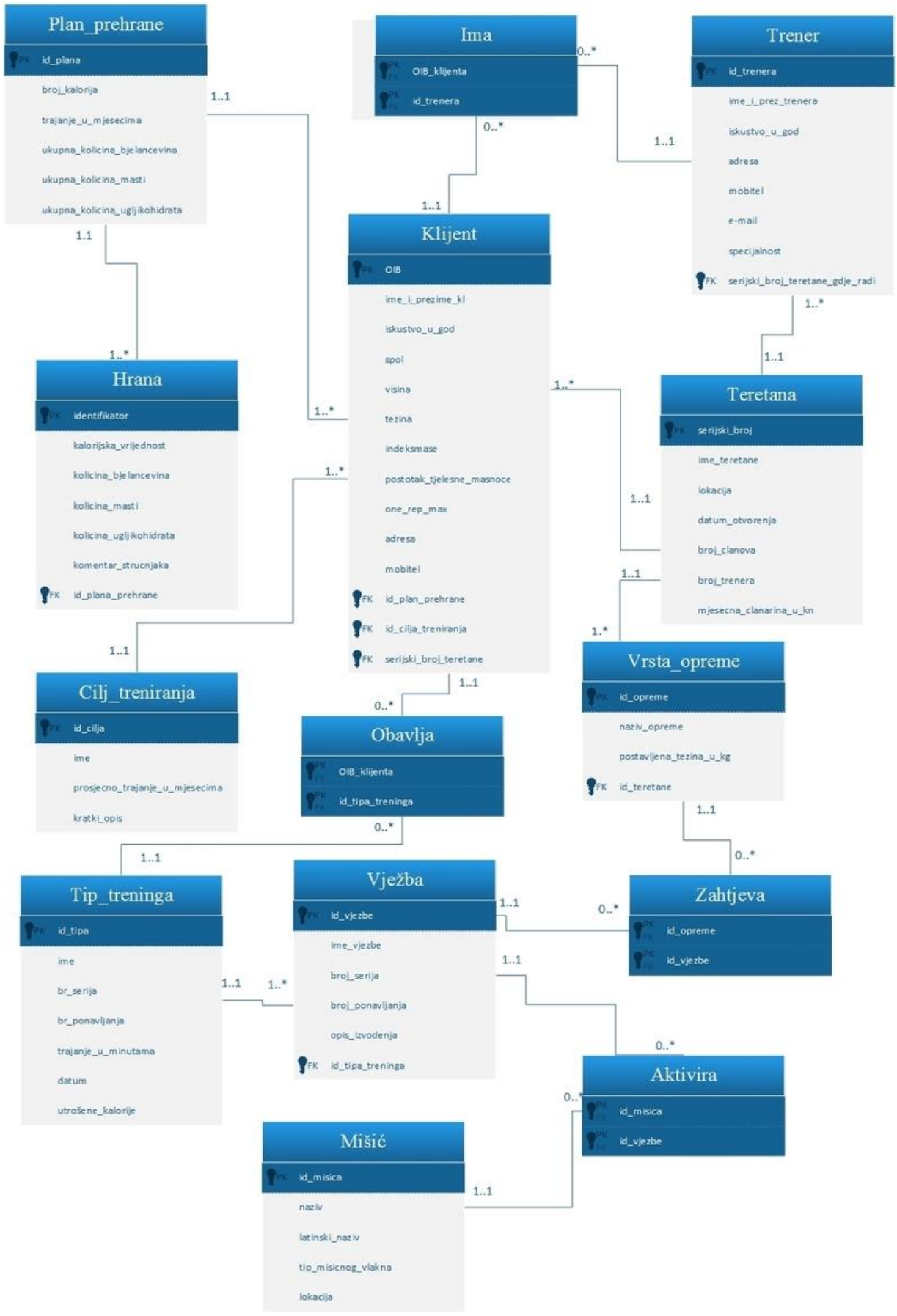
serijski_broj	Niz od 9 znakova	Jedinstveni identifikator teretane
ime_teretane	Niz od 50 znakova	Ime teretane
lokacija	Niz od 50 znakova	Fizička adresa teretane
datum_otvorenja	Datum od 1.1.1970. do 6.6.2018	Datum početka rada teretane
broj_clanova	Cijeli broj od 1 do 1500	Broj članova koji koriste usluge teretane
broj_trenera	Cijeli broj od 1 do 20	Broj trenera zaposlenih u teretani
mjesecna_clanarina_u_kn	Cijeli broj od 1 do 500	Novčani iznos mjesečne članarine u kunama
id_trenera	Niz od 10 znakova	Jedinstveni identifikator trenera
ime_i_prez_trenera	Niz od 50 znakova	Ime i prezime trenera
iskustvo_u_god	Cijeli broj od 1 do 50	Iskustvo trenera u godinama
adresa	Niz od 50 znakova	Fizička adresa trenera
mobitel	Niz od 10 znakova	Pozivni broj trenerovog mobitela
email	Niz od 50 znakova	Elektronička adresa trenera
specijalnost	Niz od 50 znakova	Dio fitnessa/sporta u kojem je trener posebno stučan

Dodatne napomene :

- navedeni rasponi atributa definirani su u Mockaroo-u (ne u SQL-u preko ograničenja), a većina tipova proizvedenih podataka mogu se smisleno proizvesti (Primjerice : Imena stvarnih ljudi, spol, fizička adresa, broj mobitela, datumi, funkcijske ovisnosti jednog atributa o drugima)

- za sve ostale atribute pri unosu nisu se koristili prikladni tipovi proizvedenih podataka jer ne postoje u Mockaroo-u (specijalnost trenera, ime teretane, naziv opreme, naziv mišića itd.), umjesto prikladnih tipova koristili su se Mockaroo tipovi Buzzword (nasumični pomodni izrazi) i Sentences (nasumične rečenice iz Lorem Ipsuma)

- ukupna količina kalorija i kalorijska vrijednost relacija Plan prehrane i Hrana dobiveni su funkcijom definiranom u Mockaroo-u : kalorijska vrijednost = (4*količina bjelančevina+4*količina ugljikohidrata+9*količina masti)



Slika 3.2. Prikaz relacijskog modela podataka Fitness sustava napravljenog u programu Microsoft Visio

3.2. Izrada relacijskog modela podataka u programu Microsoft SQL i mjerenje performansi programa

Detaljna izrada baze podataka i tablica fitness sustava u Microsoft SQL-u nalazi se u **Prilogu 1**. Shema relacijske baze napravljena je na osnovu relacijskog modela na **Slika 3.2.** . Pošto su se unosile ogromne količine podataka, strani ključevi se nisu odredili jer Microsoft SQL ne dozvoljava stvaranje n-torke sa stranim ključem koja nema svog odgovarajućeg para niti u jednoj n-torci tablice na koju se strani ključ referencira. Stvaranjem ogromne količine podataka, većina unesenih podataka neće imati svoje parove, pa će se tablice stvarati bez definiranja stranih ključeva, no atributi koji predstavljaju strane ključeve će svejedno biti prisutni. U ovom poglavlju mjerit će se brzina unosa, brzina različitih načina dohvaćanja podataka i brisanja različitog broja podataka.

3.2.1. Upiti za definiranje baze podataka Fitness sustava

Primjer 3.1. Upit za stvaranje baze podataka.

```
CREATE DATABASE Test;
```

Primjer 3.2. Upiti za stvaranje tablica Klijent i Vrsta_Opreme unutar baze podataka Test. Upiti za stvaranje ostalih tablica dani su u **Prilogu 1.1**.

```
CREATE TABLE  Klijent (
OIB VARCHAR(50)PRIMARY KEY,
ime_i_prezime_kl VARCHAR(50),
iskustvo_u_god INT,
spol VARCHAR(50),
visina DECIMAL(5,2),
tezina DECIMAL(5,2),
indeksmase DECIMAL(4,2),
postotak_tjelesne_masnoce DECIMAL(3,1),
one_rep_max INT,
adresa VARCHAR(50),
mobitel VARCHAR(50),
id_plan_prehrane VARCHAR(50) NOT NULL,
id_cilja_treniranja VARCHAR(50)NOT NULL,
serijski_broj_teretane VARCHAR(50)NOT NULL );
```

```
CREATE TABLE  Vrsta_Opreme (
id_opreme VARCHAR(50)PRIMARY KEY,
naziv_opreme VARCHAR(50),
postavljena_tezina_u_kg INT,
id_teretane VARCHAR(50) NOT NULL );
```

3.2.2. Upiti za upravljanje bazom podataka Fitness sustava

Primjer 3.3. Upiti za unošenje nasumičnih podataka stvorenih alatom Mockaroo u tablice. U primjeru se pokazuje unos 10 podataka u tablice **Misic** i **Obavlja**, a u **prilogu 1.2** je prikazan unos različitog broja podataka u sve postojeće tablice.

```
INSERT INTO Misic (id_misica, naziv, latinski_naziv, tip_misicnog_vlakna, lokacija) VALUES (80, 'workforce', 'ac nulla sed', 2, 'Ergonomic');
INSERT INTO Misic (id_misica, naziv, latinski_naziv, tip_misicnog_vlakna, lokacija) VALUES (474, 'directional', 'montes nascetur ridiculus', 2, 'toolset');
INSERT INTO Misic (id_misica, naziv, latinski_naziv, tip_misicnog_vlakna, lokacija) VALUES (143, 'multimedia', 'tortor duis', 2, 'Monitored');
INSERT INTO Misic (id_misica, naziv, latinski_naziv, tip_misicnog_vlakna, lokacija) VALUES (220, 'Upgradable', 'nunc nisl', 2, 'real-time');
INSERT INTO Misic (id_misica, naziv, latinski_naziv, tip_misicnog_vlakna, lokacija) VALUES (478, 'ability', 'lectus pellentesque', 1, 'Mandatory');
INSERT INTO Misic (id_misica, naziv, latinski_naziv, tip_misicnog_vlakna, lokacija) VALUES (287, 'Proactive', 'primis in faucibus', 1, 'Intuitive');
INSERT INTO Misic (id_misica, naziv, latinski_naziv, tip_misicnog_vlakna, lokacija) VALUES (37, 'ability', 'quis justo maecenas', 1, 'Versatile');
INSERT INTO Misic (id_misica, naziv, latinski_naziv, tip_misicnog_vlakna, lokacija) VALUES (117, 'structure', 'amet justo morbi', 1, '24 hour');
INSERT INTO Misic (id_misica, naziv, latinski_naziv, tip_misicnog_vlakna, lokacija) VALUES (737, 'help-desk', 'in', 2, 'Upgradable');
INSERT INTO Misic (id_misica, naziv, latinski_naziv, tip_misicnog_vlakna, lokacija) VALUES (13, 'zero defect', 'aliquet', 1, 'Down-sized');

INSERT INTO Obavlja (OIB_klijenta, id_tipa_treninga) VALUES ('23757594207', '91377');
INSERT INTO Obavlja (OIB_klijenta, id_tipa_treninga) VALUES ('19016631598', '80680');
INSERT INTO Obavlja (OIB_klijenta, id_tipa_treninga) VALUES ('35305907597', '94707');
INSERT INTO Obavlja (OIB_klijenta, id_tipa_treninga) VALUES ('38877375090', '95454');
INSERT INTO Obavlja (OIB_klijenta, id_tipa_treninga) VALUES ('04348126743', '76677');
INSERT INTO Obavlja (OIB_klijenta, id_tipa_treninga) VALUES ('95943236949', '12426');
INSERT INTO Obavlja (OIB_klijenta, id_tipa_treninga) VALUES ('94731677419', '29283');
INSERT INTO Obavlja (OIB_klijenta, id_tipa_treninga) VALUES ('52759472351', '25452');
INSERT INTO Obavlja (OIB_klijenta, id_tipa_treninga) VALUES ('44690952139', '28906');
INSERT INTO Obavlja (OIB_klijenta, id_tipa_treninga) VALUES ('14994774436', '18557');
```

Primjer 3.4. Složeniji upiti za dohvaćanje podataka iz tablica. U primjeru je prikazano dohvaćanje unutarnjim pridruživanjem (*eng. Inner join*) odvojenim upitima koje će dohvatiti sve vrijednosti onih n-torki dviju navedenih tablica koje imaju jednake vrijednosti navedenih atributa. Ostali upiti za dohvaćanje unesenih podataka dani su u **prilogu 1.2**.

```
SELECT * FROM Klijent Inner join Teretana on Klijent.serijski_broj_teretane=Teretana.serijski_broj;
SELECT * FROM Klijent INNER JOIN Cilj_treniranja on
Klijent.id_cilja_treniranja=Cilj_treniranja.id_cilja;
SELECT * FROM Klijent inner join Plan_prehrane on Klijent.id_plan_prehrane=Plan_prehrane.id_plana;
Select * FROM Hrana INNER JOIN Plan_prehrane on Plan_prehrane.id_plana=Hrana.id_plana_prehrane;
SELECT * FROM Vjezba INNER JOIN Tip_treninga on Vjezba.id_tipa_treninga=Tip_treninga.id_tipa;
SELECT * FROM Vrsta_Opreme INNER JOIN Teretana on
Vrsta_Opreme.id_teretane=Teretana.serijski_broj;
```

Primjer 3.5. Upiti za brisanje svih unesenih podataka iz svih definiranih tablica.

```
DELETE FROM Zahtjeva;
DELETE FROM Vrsta_Opreme;
DELETE FROM Obavlja;
DELETE FROM Ima;
DELETE FROM Trener;
DELETE FROM Hrana;
DELETE FROM Aktivira;
DELETE FROM Klijent;
DELETE FROM Vjezba;
DELETE FROM Tip_treninga;
DELETE FROM Misic;
DELETE FROM Teretana;
DELETE FROM Plan_prehrane;
DELETE FROM Cilj_treniranja;
```

3.2.3. Mjerenje performansi MSSQL baze podataka

U ovom potpoglavlju mjerit će se vremena potrebna za : unos ogromne količine podataka, dohvaćanje svih podataka svih tablica, dohvaćanje svih podataka tablica preko 4 oblika pridruživanja (unutarnje pridruživanje odvojenim upitima, unutarnje pridruživanje svih vezanih relacija, unutarnje pridruživanje vezanih relacija preko definiranih indeksa na tablicama i potpuno vanjsko pridruživanje), te brisanja svih podataka tablice. Unosit će se 10, 50, 100, 500, 1000, 5000 i 10000 podataka svake relacije, osim relacije **Misic** i **Aktivira** koji mogu imati samo 752 različite n-torke (752 primarna ključa). Njih će se unositi 752 kad unos podataka u svaku tablicu bude veći od 500 jer će Mockaroo moći proizvesti svega 752 različite n-torke tih relacija (mišića u ljudskom tijelu ima 752). Alat koji će se koristiti za mjerenje vremena obavljanja navedenih operacija bit će **Query Client Statistics** unutar MSSQL-a. On će pokazati ukupno vrijeme u milisekundama koje je potrebno da se izvedu sve operacije navedene u SQL-ovom prostoru za upite. To vrijeme zbroj je **klijentovog vremena obrade** (*eng. Client processing time*) i **vremena čekanja na odzive SQL servera** (*eng. Wait time on server replies*). Klijentovo vrijeme za obradu poslanih upita u potpunosti ovisi o radnom taktu procesora klijentovog računala, a vrijeme čekanja na odzive SQL servera ovisi o MSSQL-u. Pri mjerenjima performansi bit će navedena oba parametra. Kao što je navedeno u **poglavlju 2.3**, Mockaroo

proizvođač nasumičnih podataka dopušta proizvodnju od najviše 1000 podataka za svaku tablicu (relaciju) odjednom (korištenje besplatne verzije alata Mockaroo), tako da se stvaranje svih n-torki svake relacije (osim **Misic** i **Aktivira**) vršilo 5 puta (5*1000) za 5000 podataka svake relacije i 10 puta (10*1000) za 10000 podataka svake relacije. Tijekom tog procesa dolazilo je do ponavljanja primarnih ključeva n-torki, a sve n-torke unutar tablice moraju imati jedinstven primarni ključ jer je to atribut koji jedinstveno određuje n-torku. Zbog toga je uvedena varijabla **Uspješnost unosa** koja je omjer uspješno unesenih n-torki u tablice (***Rows affected by INSERT, UPDATE or DELETE statements*** dio statistike upita) i ukupnog broja upita za unos n-torki u tablice (***Number of INSERT, UPDATE or DELETE statements*** dio statistike upita). Neuspješno unesene n-torke imaju iste primarne ključeve kao prethodno unesene n-torke, a MSSQL neće dopustiti unos dva ista podatka. U nastavku je prikazana konfiguracija računala na kojemu se vršilo testiranje (konfiguracija određuje **Klijentovo vrijeme za obradu upita**) i tablica performansi za svaki različit broj unesenih podataka u tablice.

Procesor : Intel(R) Celeron(R) CPU N2840 2.16 GHz Instalirana memorija (RAM) : 4 GB (3.89 iskoristivo) Vrsta sustava : 64-bitni operacijski sustav, procesor x64

Slika 3.3. Konfiguracija računala na kojem su se vršila ispitivanja performansi

Tablica 3.2. Rezultati mjerenja navedenih parametara MSSQL Servera izraženi u milisekundama (U zagradama je navedeno **Klijentovo vrijeme obrade** + **Vrijeme čekanja na odziv servera**) :

Broj podataka svake tablice	**Ukupan broj podataka**	**Unos podataka (klijent + server)**	**Dohvaćanje svih podataka (klijent + server)**	**Unutarnje pridruživanje odvojenim upitima (*eng. Inner Join*)**	**Unutarnje pridruživanje svih relacijski povezanih tablica odjednom**	**Unutarnje pridruživanje svih povezanih tablica preko definiranih indeksa**	**Potpuno vanjsko pridruživanje (*eng.full outer join*)**	**Brisanje svih podataka tablica**	**Uspješnost unosa(%)**
10	140	770(739+31)	78 (78+0)	146(65+81)	259(92+167)	234(78+156)	100(97+3)	62(47+15)	100
50	700	1999(1990+9)	182 (158+24)	180(40+140)	100(100+0)	171(156+15)	171(115+56)	195(65+130)	100
100	14000	2765(2156+609)	218 (109+109)	230(79 + 151)	140(94+46)	125(94+31)	175(171+4)	212(75+137)	100
500	7000	8969(8016+953)	328(250+78)	384(72+312)	861(106+755)	204(126+78)	388(340+48)	265(47+218)	100
1000	13504	28808(25762 +3046)	343(343+0)	414(110+304)	1005(128+877)	278(125+153)	315(284+31)	1125(79+1046)	100
5000	61504	206025(73144+132881)	1047(938+109)	1157(149+1008)	1484(250+1234)	1083(302+781)	1553(1410+143)	2859(62+2797)	77,715
10000	121504	1959053(785302+1173752)	4172(3922+250)	1668(220+1448)	3312(343+2969)	2328(407+1921)	4252(2870+1382)	7734(140+7594)	75,3

3.3. Unos u Redis nerelacijsku bazu podataka i mjerenje performansi baze

U ovom potpoglavlju opisat će se unos, dohvaćanje i brisanje ogromne količine podataka u Redis bazi podataka. Kao što je opisano u uvodu, unos podataka će se vršiti preko okruženja Visual Studio, odnosno preko konzolne aplikacije u programskom jeziku **C#,** dohvaćanje će se mjeriti preko **Benchmarka** nakon što su podaci uneseni (GET time), a brisanje podataka izvest će se preko Redis komandne linije (*eng. Redis Command Line Interface*). Za mjerenje performansi servera koristit će se alat **Redis-Benchmark**, a za jednostavniji pregled svih unesenih tablica (u obliku Redis hasheva) će se koristiti alat **Redis Desktop Manager**.

3.3.1. Opis rada s Redis serverom preko C# konzolne aplikacije

Za spajanje konzolne aplikacije na Redis Server nužno je instalirati NuGet paket **StackExchange.Redis** koji sadrži biblioteku s naredbama za manipulaciju nad serverom. Nakon preuzimanja i instalacije NuGet paketa potrebno je navesti korištenje navedene biblioteke (using **StackExchange.Redis**). Spajanje s bazom podataka ostvaruje se postavljanjem veze preko klase **ConnectionMultiplexer** i njezinih funkcija **Connect**(„ime_veze“) i **GetDatabase**(), te preko tipa podatka **IDatabase**. Za početak potrebno je definirati klase čiji će objekti predstavljati n-torke relacije Fitness sustava. To su klase : Klijent, Plan_prehrane, Hrana, Tip_treninga, Cilj_treniranja, Misic, Vjezba, Vrsta_Opreme, Teretana i Trener. Za modeliranje klasa nije potrebno koristiti relacijski model s entitetima-vezama i stranim ključevima, pošto baza podataka nije relacijska. Atributi klasa definirani su u 3.1 preko odgovarajućih tipova podataka u C# programskom jeziku. U konstruktorima klasa (metode koje postavljaju predodređene vrijednosti objektima) se koristila klasa Random iz standardne biblioteke System koja je omogućila postavljanje nasumičnih vrijednosti atributa svakog objekta po rasponima danim u 3.1. Za potrebe postavljanja predodređenih (*eng.default*) vrijednosti atributa koji su decimalni brojevi koristila se javna funkcija **GetRandomDouble** koja nije definirana u standarnoj biblioteci (definirana je naknadno). Definiranjem objekata u glavnom dijelu programa i dohvaćanjem njihovih atributa u strukturi **HashEntry[]** određuju se vrijednosti polja Redis Hasha, a ključ se definira naredbom **HashSet(varijabla, HashEntry)**. Izvedbom konzolne aplikacije podaci se unose u Redis bazu podataka.

Primjer 3.6. Definiranje veze s bazom podataka preko klasa, funkcija i tipova podataka iz biblioteke StackExchange.Redis.

```
ConnectionMultiplexer redisconn = ConnectionMultiplexer.Connect("localhost");
IDatabase redDB = redisconn.GetDatabase();
```

Funkcija **Connect** kao parametar uzima ime baze podataka na koju se spaja. U varijablu **redDB** naredbom **GetDatabase()** spremila se Redis baza podataka pod imenom localhost. Preko definirane varijable je tako moguće vršiti operacije s bazom preko operatora točka.

Primjer 3.7. Definiranje nasumičnih vrijednosti atributa klase Klijent u određenim rasponima

```
public Random  r = new Random((int)DateTime.Now.Millisecond);//stvaranje svake milisekunde
iskustvo_u_godinama = r.Next(0, 50);//raspon  od 0 do 50
visina_cm = r.Next(150, 220);
tezina_kg = r.Next(50, 150);
indeksmase = GetRandomDouble(r, 18.8, 45.2);
postotak_tjelesne_masnoce = r.Next(5, 40);
one_rep_max = r.Next(50, 250);
```

Primjer 3.8. Definiranje nasumičnih vrijednosti atributa klase Klijent predstavljenih kao polje znakova tako što se uzimaju nasumični elementi iz prethodno definiranih vektora (slova, brojevi i spolovi).

```
public Random  r = new Random((int)DateTime.Now.Millisecond);
private char[] spolovi = "MZ".ToCharArray();
private char[] slova = "abcdefghijklmnopqrstuwvqxyz ".ToCharArray();
public char[] brojevi = "1234567890".ToCharArray();
for (int i = 0; i < 20; i++)
{   this.ime_i_prezime_kl += slova[r.Next(0, 28)];
    this.adresa += slova[r.Next(0, 28)];}
for (int i = 0; i < 11; i++)
 {  this.OIB += brojevi[r.Next(0, 10)];}
for (int i = 0; i < 9; i++)
 {  this.mobitel += brojevi[r.Next(0, 10)];}
```

Brojevi do kojih ide for petlja je broj nasumičnih elemenata koji se odabire iz prethodno definiranih vektora, a broj naveden kao parametar funkcije **Random.Next** je raspon elemenata polja (slova, brojevi ili spolovi) koji se uzimaju za dodavanje u atribute. Sve definirane klase dane su u **prilogu 2.1**.

Primjer 3.9. Definiranje objekta Vrsta_Opreme iz definiranih klasa, stvaranje varijable za spremanje u bazu podataka koja će predstavljati ključ, definiranje hashovih atributa, te unos hasha u bazu.

```
Vrsta_Opreme vo1 = new Vrsta_Opreme(); //objekt vo1 kao instanca klase
var  k91 = "Vrsta_Opreme1";  //Varijabla koja sadrži ime ključa
HashEntry[] vrsta_opremedetalji1 = {
        new HashEntry("id_opreme",vo1.id_opreme),
        new HashEntry("naziv",vo1.naziv),
        new HashEntry("postavljena_tezina_kg",vo1.postavljena_tezina_kg)
}; //definirani atributi podatka bit će isti oni koji su nasumično generirani u klasi
redDB.HashSet(k91, vrsta_opremedetalji1); //unos hasha s definiranim atributima u bazu podataka
```

Kodovi za unos različitih količina podataka dani su u **prilogu 2.1**.

3.3.2. Opis mjerenja performansi

Mjerenje vremena unosa mjerit će se preko alata Redis servera zvanog **Redis Benchmark**. Redis Benchmark je alat koji može mjeriti i simulirati operacije nad podacima, te kao rezultate prikazati vremena potrebna za obavljanje danih operacija. Redis Benchmark može se koristiti na dva načina : **neizravno pratiti izvještaje Benchmarka** koji govore koliko vremena je potrebno za izvođenje svih operacija (posebno za operacije GET, SET, INCR itd.) koje server trenutno obavlja (izvještaji svakih 10-15 sekundi), te **izravno** ulančavanjem naredbi preko naredbenog retka. Izravno mjerenje je simulacija brzine odziva servera na dane upite. Preko izravnog mjerenja Redis benchmark simulira naredbe dane od N klijenata za M upita, te izvještava korisnika o trajanju izvođenja upita pri danim uvjetima. **Izravno mjerenje** je točnije od **neizravnog** jer se preko neizravnog praćenja dodatno uračunava vrijeme potrebno za očuvanje konzistentnosti podataka servera i ostale operacije održavanja podataka na Redis serveru. Unos podataka preko C# konzolne aplikacije dio je **neizravnog mjerenja** Redis performansi.

Primjer 3.10. Opća sintaksa za izravno testiranje trajanja upita preko naredbenog retka određenog operacijskog sustava

```
redis-benchmark [-h <host>] [-p <port>] [-c <clients>] [-n <requests>] [-k <boolean>]
```

Primjer 3.11. Upit za prikaz performansi pri postavljanju 100 nasumičnih redis hasheva

```
redis-benchmark -r 100 -n 100 HMSET -q
```

Dani upit govori Redis Benchmarku da simulira unos 100 nasumičnih podataka za unos, te da postavi 100 zahtjeva dok to čini. HMSET naredba služi za dodavanje više hasheva u bazu

podataka. Oznaka –q govori benchmarku da kao rezultat izbacuje samo vrijednosti upita po sekundi. Pošto se u naredbi nije navela veličina svakog podatka i broj veza sa serverom, benchmark pretpostavlja da se upiti šalju preko 50 različitih veza (*eng. Connections*) i da je svaki uneseni podatak veličine 3 bajta.

Rezultat upita : 100 zahtjeva obavljeno u 0.05 sekundi
50 paralelnih veza (klijenata)
Podaci veličine 3 bajta

3.3.3. Mjerenje performansi Redis baze podataka

Tablica 3.3. Rezultati mjerenja navedenih parametara u milisekundama preko izravno (simulacijom) preko alata Redis-Benchmark. Važno je naglasiti da Redis nema mogućnosti mjerenja veličine pojedinog unešenog podatka, pa se tako ne može saznati koliko će podaci Fitness sustava biti veliki i napraviti veoma preciznu simulaciju unosa, dohvaćanja i brisanja podataka preko benchmarka. Upiti koji su se koristili za simulaciju dani su u tekstnom dokumentu u **prilogu 2.2**.

Ukupan broj podataka	Unos n-torki (naredba HMSET)	Dohvaćanje svih n-torki s atributima i njihovim vrijednostima(naredba HGETALL)	Dohvaćanje vrijednosti atributa svih n-torki (naredba HVALS)	Brisanje svih unešenih podataka tablice (naredba FLUSHALL)
100	20 ms	20 ms	50 ms	50 ms
500	130 ms	110 ms	120 ms	110 ms
1000	210 ms	220 ms	180 ms	210 ms
5000	960 ms	630 ms	700 ms	640 ms
10000	1170 ms	1180 ms	1150 ms	1190 ms
50000	6340 ms	5820 ms	5790 ms	6350 ms
100000	12710 ms	11600 ms	11810 ms	11910 ms

Tablica 3.4. Rezultati mjerenja navedenih parametara u milisekundama neizravno preko Redis-benchmarka. Unos podatka radio se preko konzolne aplikacije, dohvaćanje i inkrementiranje se mjerilo Benchmarkom nakon unosa podataka, a brisanje preko Redis naredbenog retka (*eng. Redis Command Line Interface*). Nakon unosa određenog broja podataka preko C# konzolne aplikacije, **Redis-Benchmark** će računati i prikazati vrijeme potrebno za operacije s podacima. U danoj tablici su rezultati mjerenja za operacije SET (postavljanje vrijednosti), GET (dohvaćanje vrijednosti), INCR (uvećavanje vrijednosti ključa koji sadrži brojčanu vrijednost) i FLUSHALL (brisanje svih ključeva).

Broj podataka svake relacije	Ukupan broj podataka	Unos podataka (SET)	Dohvaćanje svih podataka (GET)	Uvećavanje vrijednosti ključa s brojčanom vrijednosti (INCR)	Brisanje svih podataka (FLUSHALL)
10	100	134 ms	54 ms	50 ms	67 ms
50	500	264 ms	157 ms	44 ms	84 ms
100	1000	356 ms	278 ms	445 ms	92 ms
500	5000	882 ms	672 ms	3395 ms	108 ms
1000	10000	4446 ms	1568 ms	2055 ms	1767 ms
5000	50000	7110 ms	5095 ms	4454 ms	5056 ms
10000	100000	16370 ms	13450 ms	15930 ms	12590 ms

3.4. Usporedba MSSQL-a i Redisa

U tablici je dana usporedba dviju bazi podataka preko kojih se radila poslovna aplikacija.

Tablica 4.1. Usporedba MSSQL-a i Redisa na temelju njihovih osobina

Osobina	Microsoft SQL	Redis
Struktura baze	Relacijska baza podataka, podaci su raspoređeni po tablicama.	Upis podataka u memoriju preko osnovne strukture (ključ) i njezinih inačica.
Svrha baze	Baza podataka za jednostavan unos i pretraživanje u memoriji.	Baza podataka, skladište podataka predmemorije (*eng. cache*), posrednik za poruke.
Programski jezik implementacije	C++	C
Transakcije	Podržava transakcije, pri čemu zadržava ACID svojstva (nedijeljivost, dosljednost, izolaciju i izdržljivost) svojih podataka.	Transakcije kao skup upita koji se izvode između naredbi MULTI i EXEC, atomsko i veoma brzo izvršavanje naredbi.
Tipovi podataka	Strukturirani tipovi podataka, točno definirani rasponi vrijednosti.	Nestrukturirani tipovi podataka, u ključeve se mogu spremati sve vrijednosti bez prethodnog navođenja tipa.
Aplikacija preko koje se pristupa bazi podataka	Microsoft SQL Server Manager Studio.	Redis naredbeni redak, Redis Desktop Manager kao dodatna aplikacija za bolju preglednost i lakše mijenjanje vrijednosti podataka.
Operacijski sustavi	Windows, Linux.	Windows, Linux, OS X, BSD.
Unos podataka	Posebno definiranje preko DDL upita, naknadni unos vrijednosti podataka po definiranoj relaciji preko DML upita.	Definiranje ključa i spremanje vrijednosti jednim upitom, moguće naknadne promjene vrijednosti.

Predstavljanje objekta (n-torke)	Redak definirane tablice je skup atributa jedne n-torke.	Moguće jedino preko Hash ključa i njegovih polja koji predstavljaju atribute n-torke.
Manipularanje bazom podataka	Moguće dohvaćanje, brisanje i mijenjanje podataka preko korisnički definiranih selekcija (Primjerice, želi se dohvatiti sve korisnike kojima je ime Ivo).	Nije moguća selekcija pri upitu, ako se želi dohvatiti,obrisati ili promijeniti podatak koji neki ključ (polje) sadrži, moramo navesti ime ključa.
Sigurnost i kontrola pristupa	Omogućuje korisničku autentikaciju preko Microsoft računa (korisničko ime i lozinka) ili preko samog operacijskog sustava.	Jednostavan pristup serveru preko lozinke ukoliko je korisnik servera postavi.
Brzina obrade upita	Prihvatljiva brzina obrade upita, ovisi o brzini slanja upita (klijentovo računalo) .	Najbrža postojeća baza podataka, pojednostavljuje aplikacijski model.
Ključevi	Postoje, određuju jedinstvenu n-torku definirane relacije.	Nije ih moguće definirati, strukture podataka skupovi i poredani skupovi ne dozvoljavaju više istih vrijednosti (slična uloga kao ključevi). U ostalim strukturama podataka nije moguće ograničiti jedinstvenost atributa preko kandidat-ključa
Programski jezici koje baza podataka podržava	C#, C++, Go, Java, JavaScript, PHP, Python, Ruby itd.	Podržava više programskih jezika nego SQL među kojima su : C#, C, C++, Haskell, Matlab, Swift, Visual Basic itd.

3.5. Interpretacija dobivenih rezultata

U ovom potpoglavlju slijedi interpretacija rezultata u tablicama **3.2**, **3.3** i **3.4** dobivenih u poglavlju 3. Prilikom rada s SQL Serverom brzo su se definirale tablice u kojih će se unositi n-torke čiji su atributi nasumično proizvedeni Mockaroo-om. Kao što je očekivano, povećanjem broja unesenih podataka raste i vrijeme potrebno za izvođenje danih upita. Za mal broj podataka SQL server iznimno brzo obrađuje upite, puno kraće nego klijent. Kod unosa velikog broja podataka (5000 i 10000) serverovo vrijeme za obradu podataka postaje veće od klijentovog.

Dohvaćanje svih podataka traje nešto duže od dohvaćanja podataka preko unutarnjeg pridruživanja jer postoji malen broj n-torki kojima je dani atribut (preko kojeg se n-torke spajaju) iste vrijednosti (prvo preklapanje tek nakon unosa 500 n-torki svake relacije). Kod dohvaćanja podataka (kao i kod brisanja) serverovo vrijeme obrade upita je stalno veće od klijentovog jer klijent kod dohvaćanja (brisanja) samo šalje upite, a server obrađuje upite i pretražuje bazu po danim kriterijima. Unutarnje pridruživanje u kojem su se spajale sve tablice vezane ključevima odjednom je kraće trajalo od unutarnjeg pridruživanja s više upita, što pokazuje da će vrijeme izvođenja upita biti nešto kraće ako je upita manje. Nakon postavljanja indeksa na kandidat-ključeve (primarne i strane ključeve) tablica, dohvaćanje svih relacijski vezanih podataka tablica odjednom, traje otprilike jednako kao i bez definiranih indeksa pri manjem broju podataka, no pri većem broju podataka (1000 podataka svake relacije pa nadalje) uvelike se smanjuje vrijeme dohvaćanja podataka iz tablica. Dobiveni rezultat pokazuje funkcionalnost indeksa u SQL bazama podataka - pretraživanje baze podataka s definiranim indeksima je puno brže. Performanse Redis servera izmjerene posredno (simuliranjem izvođenja naredbi) i neposredno (preko C# konzolne aplikacije) dosta se razlikuju. Razlika je očekivana pošto rezultati simuliranih operacija govore brzinu izvođenja upita bez uračunatog vremena koje je serveru potrebno za održavanje konzistentnosti podataka. Vremena koja su potrebna za izvođenje upita na Redis serveru su puno su manja od vremena potrebnih za izvođenje upita u SQL-u koji imaju istu (ili sličnu) svrhu.

4. ZAKLJUČAK

Poslovna aplikacija Fitness sustava puno se lakše predstavlja preko relacijskog modela podataka, odnosno korištenjem relacijske SQL baze podataka. Iako Redis kao baza podataka uvelike smanjuje složenost aplikacije, te brže izvodi operacije s podacima, još uvijek ima veliki problem kod dohvaćanja podataka (dohvaćanje n-torki sa željenim atributima po navedenim kriterijima nije moguće). Također, Redis kao „open-source“ ima puno ograničenja kod mjerenja zauzete memorije i performansi (dosta neprecizno mjerenje), za razliku od RedisLabs plative verzije za veća udruženja. Redis serverom je pogodnije manipulirati preko neke druge platforme (drugog programskog jezika) jer glavno sučelje (Redis naredbeni redak) nije pogodno ukoliko se treba izvesti više odvojenih upita odjednom. Istina je da SQL Server Management Studiu treba puno više (i do 10 puta više) vremena za izvođenje operacija, ali ta vremena nisu toliko velika da bi se tražila druga brža baza podataka. SQL daje puno veću mogućnost manipulacije podacima (ograničenja i ključevi) i omogućuje smanjenje vremena dohvaćanja podataka preko indeksa u slučaju pretraživanja veće količine podataka. Za izradu, kao i za korištenje, prednost kod poslovnih aplikacija ima SQL. Poslovna aplikacija Fitness sustava lakša je za napraviti u SQL-u, te je jednostavnija za korištenje trenerima, klijentima i službenicima teretane. Redis bi svoju svrhu našao u evidentiranju ulazaka u teretanu, evidentiranju dužnika, praćenju aktivnosti korisnika u teretani, vođenju tablice bodova (*eng. leaderboards*) za određena natjecanja i rekorde u teretani i sl.

LITERATURA

[1] **Beginning Database Design; Claire Churcher; 2012. godine** – Modeliranje podataka, ER model podataka, relacijski model podataka, pretrvaranje ER modela podataka u relacijski model; 11.5.2018-24.5.2018.

[2] **Programiranje SQL Server 2005; Bill Hamilton; 2005. godine** – SQL jezik (osobine i podjela), SQL Server Management Studio; 11.5.2018-24.5.2018.

[3] **Redis Cookbook; Tiago Macedo, Fred Oliveira; 2011. godine** – Redis (osobine i strukture podataka); 10.4.2018.

[4] **JavaTpoint** https://www.javatpoint.com/redis-tutorial - Osobine Redisa, strukture podataka, upiti; 17.5.2018.

[5] **Mockaroo Random Data Generator and API Mocking Tool** https://www.mockaroo.com/help/about – Osobine i razvoj Mockaroo-a; 4.6.2018.

[6] **SQLServerCentral** http://www.sqlservercentral.com/articles/Performance+Tuning/measuringperformance/1323/- Načini mjerenja performansi SQL servera, klijentovo vrijeme obrade, vrijeme čekanja na odziv servera; 7.5.2018.

[7] **SQLServer.info** http://www.sqlserver.info/management-studio/show-query-execution-time/ - Mjerenje izvođenja upita; 30.5.2018.

[8] **YouTube, kanal DevNami** https://www.youtube.com/watch?v=KvQJFzI765c – Spajanje C# konzolne aplikacije na Redis bazu podataka; 23.5.2018.

[9] **Tashwar Bhatti, programerski blog** http://taswar.zeytinsoft.com/redis-hash-datatype/ - Manipulacija Redis Hash strukuturom podataka preko C# konzolne aplikacije; 28.5.2018.

[10] **YouTube, kanal thenewboston** https://www.youtube.com/watch?v=U9RdDVstjM8 – Generiranje nasumičnog niza znakova u C# klasama za unos u Redis bazu podataka; 26.5.2018.

[11] **DB Engines** https://db-engines.com/en/system/Microsoft+SQL+Server%3BRedis – Kriteriji za uspoređivanje dviju baza podataka; 23.7.2018.

[12] **Redis.io** https://redis.io/topics/benchmarks – Mjerenje performansi Redis servera, opis načina rada alata Redis-Benchmark; 7.6.2018.

SAŽETAK

Glavni cilj završnog rada bio je usporediti relacijsku Microsoft SQL bazu podataka i nerelacijsku Redis bazu podataka na primjeru poslovne aplikacije Fitness sustava. Cilj aplikacije je vođenje klijentovih podataka o zdravom životu i tjelovježbi. Aplikacija također služi trenerima i službenicima teretane za nadziranje podataka o postojećim klijentima. Aplikacija je svojevrstan posrednik klijenata i davatelja fitness usluga (djelatnika teretane i trenera). U radu su opisane osnovne osobine i strukture podataka SQL-a i Redisa, te osnovne osobine alata Mockaroo nasumičnog proizvođača podataka i C# programskog jezika preko kojih su se podaci aplikacije unosili u odgovarajuće baze. Također je opisan alat Microsoft Visio preko kojeg se radilo modeliranje podataka za relacijsku (SQL) bazu podataka. Nakon unosa varijabilne količine podataka, dvije baze podataka uspoređivale su se po : brzini unosa, pristupa i brisanja podataka, te preglednosti i sigurnosti unesenih podataka.

Ključne riječi : Microsoft SQL, Redis, Relacijski model podataka, C# programski jezik, Mockaroo proizvođač stvarnih podataka, Microsoft Visio, No-SQL

ABSTRACT

The comparison of Redis and MSSQL databases on Bussiness application

The papers main goal was the comparison of relational Microsoft SQL database and non-relational Redis database on Bussiness application of Fitness system model. An application's goal is a maintenance of clients information about healthy living and workout. The Application is also used for coach and gym manager supervision of existing clients data. The Application is an intermediator between clients and fitness services givers (gym officials and coaches). Redis and MSSQLs main features and data structures are described in the paper, as well as features of Mockaroo realistic data generator and C# programming language which were used for data insertion in each database. Microsoft Visio features, the tool used for data modeling for the relational database (SQL) were also described. After the mass data insertion, two databases were compared by the following performances : insertion, fetching and deletion of data as well as database transparency and security.

Keywords : Microsoft SQL, Redis, Relational database model, C# programming language, Mockaroo realistic data generator, Microsoft Visio, No-SQL

ŽIVOTOPIS

Filip Ćuić rođen je 17. svibnja 1996. godine u Našicama. Od rođenja živi u Našicama, gdje stječe osnovnoškolsko obrazovanje. Godine 2011. upisuje Srednju Školu Izidora Kršnjavoga Našice, smjer Opća Gimnazija. Sve razrede od početka do kraja obaveznog školovanja prolazi s odličnim uspjehom. Školske godine 2014./15. završava srednju školu, te akademske godine 2015./16. upisuje prvu godinu preddiplomskog studija računarstva na Elektrotehničkom Fakultetu Osijek koji godinu kasnije mijenja ime u Fakultet elektrotehnike, računarstva i informacijskih tehnologija. Navedeni fakultet još uvijek pohađa.

PRILOZI

[1] **Prilog 1.1** DDL upiti, upiti za definiranje i brisanje tablica u SQL-u, upiti za definiranje tablica generirani pomoću Mockaroo-a

[2] **Prilog 1.2** DML upiti, upiti za dodavanje, dohvaćanje i brisanje n-torki tablica

[3] **Prilog 2.1** Upiti za unos podataka u Redis bazu podataka preko C# konzolne aplikacije

[4] **Prilog 2.2** Upiti za simulaciju operacija Redis servera preko naredbenog retka operacijskog sustava