Sustav za upravljanje bazama podataka MariaDB

Ilišević, Irena

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Irena Ilišević

SUSTAV ZA UPRAVLJANJE BAZAMA PODATAKA MARIADB

ZAVRŠNI RAD

Varaždin, 2019.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Irena Ilišević

Matični broj: 44945/16–R

Studij: Informacijski sustavi

SUSTAV ZA UPRAVLJANJE BAZAMA PODATAKA MARIADB

ZAVRŠNI RAD

Mentor:

Prof. dr. sc. Kornelije Rabuzin

Varaždin, srpanj 2019.

Irena Ilišević

Izjava o izvornosti

Izjavljujem da je moj završni rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristila drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autorica potvrdila prihvaćanjem odredbi u sustavu FOI-radovi

_____

Sažetak

Kako sam naslov teme upućuje, u radu će biti riječ o sustavu za upravljanje bazama podataka MariaDB. Dakle, biti će opisane osnovne karakteristike tog sustava, odnosno pojašnjene osnove rada s tim sustavom od kreiranja baze podataka, kreiranja, brisanja, ažuriranja tablica i podataka u njima, rad s virtualnim stupcima i okidačima. Osim toga, napraviti ću usporedbu sustava za upravljanje bazama podataka MariaDB te MySQL. Navesti glavne prednosti i nedostatke tog sustava te slučaje korištenja.

U praktičnome dijelu rada na primjeru baze podataka za potrebe autoškole prikazati ću rad sa sustavom za upravljanje bazama podataka MariaDB. Pokazati osnovne karakteristike sustava na primjeru. Uz to pokazati ću kako koristiti taj sustav u kombinaciji s alatom DataGrip.

Ključne riječi: sustav za upravljanje bazama podataka; baze podataka; MariaDB; modeliranje; DataGrip; relacijske baze podataka.

Sadržaj

1. Uvod

Kako su baze podataka sastavni dio skoro svakoga programskoga rješenja, tako su one i neizostavan dio mog studiranja. Osim proučavanja samih baza podataka od ključne su važnosti sustavi za upravljanje bazama podataka kojih je trenutno mnogo na informatičkom tržištu. Svaki od njih ima vlastite prednosti i nedostatke te, dakako, razloge zašto ih koristiti. Nekima je prednost brzina, nekima pohrana velikih količina podataka, nekima sigurnost itd. Budući da sam se kroz studiranje bavila sustavima za upravljanje bazama podataka poput PostgreSQL-a, MS SQL Servera, Oracle-a i MySQL-a koji su prisutni dugo godina na današnjemu tržištu, zanimalo me kako funkcioniraju neki novi sustavi i neke nove ideje u svijetu baza podataka. Nakon nekoga vremena istraživanja utvrdila sam da je najbrže rastući novi sustav za upravljanje bazama podataka upravo MariaDB.

Ovaj sustav za upravljanje bazama podataka ističe se svojim performansama te različitim mogućnostima koje nudi svojim klijentima, gdje između ostaloga klijent može birati između dosta različitih podsustava koje može integrirati u svoje rješenje. Osim toga, MariaDB nudi mogućnost korištenja dinamičkih i virtualnih stupaca, pohranjivanje geometrijskih i geoprostornih podataka, kreiranje mnoštva različitih indeksa i još mnogo toga. U današnjemu modernome svijetu gdje se podaci velikom brzinom mijenjanju i sve ih je više, veliki je izazov pred bazama podataka kako bi se svi podaci uspješno pohranili. No, danas je mnogo sustava za upravljanje bazama podataka koji omogućavaju uspješnu pohranu velike količine podataka, ali nisu svi baš tako uspješni u njihovom brzom pretraživanju. Upravo, MariaDB nudi mogućnost naprednoga pretraživanja teksta.

Dakle, postoji dosta razloga za proučavanje ovoga sustava za upravljanje bazama podataka pa u skladu s time u radu ću najprije nešto reći o samim bazama podataka, te poziciji sustava za upravljanje bazama podataka MariaDB u svijetu baza podataka, zatim ću iznijeti osnovne crte nastanka ovoga sustava, osnovne karakteristike, načine rada, pokazati povezivanje s nekim drugim alatima koji olakšavaju rad te na primjeru baze podataka pokazati njegove karakteristike.

2. Metode i tehnike rada

U izradi ovoga Završnoga rada služit ću se sljedećim alatima: sustavom za upravljanje bazama podataka MariaDB i to verzijom 10.3.14-GA koja je za besplatno preuzimanje dostupna na službenim stranicama. Osim MariaDB Servera, upotrebljavati ću i alat JetBrains DataGrip 2018 koji je dostupan preko službenih stranica i za koji posjedujem studentsku licencu.

Sam rad je strukturiran tako da će u prvome dijelu biti iznesena teorija vezana uz baze podataka gdje ću ukratko objasniti karakteristike relacijskih i nerelacijskih baza podataka. Zatim će biti definiran pojam sustava za upravljanje bazama podataka i pojam sheme baze podataka. Nakon uvoda u baze podataka slijedi uvod u sustav za upravljanje bazama podataka MariaDB gdje će biti prikazan nastanak samoga SUBP MariaDB. U sklopu ovoga poglavlja biti će iznesene osnovne karakteristika SUBP MariaDB, odnosno biti će riječ o upitnim jezicima koji se koriste, o osnovnim relacijskim karakteristikama te o radu pomoću SQL-a kao jezika. Osim toga, biti će riječ i o nerelacijskim karakteristikama koje podržava SUBP MariaDB. Nakon osnovnih karakteristika, slijede naprednije karakteristike poput naprednog pretraživanja teksta, rada s geometrijskim i geoprostornim podacima, rad s virtualnim i dinamičkim stupcima i dr. Nakon definiranja osnovnih karakteristika SUBP MariaDB, biti će navedene osnovne prednosti i nedostatci ovoga sustava. Radi boljega prikaza pozicije sustava za upravljanje bazama podataka MariaDB u svijetu baza podataka, slijedi, poglavlje u kojemu će biti usporedba SUBP MariaDB sa SUBP MySQL.

U praktičnome dijelu rada karakteristike sustava za upravljanje bazama podataka MariaDB biti će prikazane kroz bazu podataka za potrebe autoškole. Tu će najprije biti riječ o kreiranju baze podataka i strukturiranju tablica. Nakon toga, slijedi implementacija ograničenja, prikaz postavljanja jednostavnih i složenih upita, kreiranje okidača te rad s virtualnim stupcima i naprednim pretraživanjem teksta.

Radi boljeg i potpunijeg shvaćanja i razumijevanja činjenica, podataka i statistika vezanih u SUBP MariaDB, radu sam dodala raznovrstan didaktički materijal (razne slike, upute, tablice). Kod pripreme rada nailazila sam na razne probleme poput teškoga pronalaska relevantnih izvora literature. Građu (materijal) o povijesti i karakteristikama SUBP MariaDB nastojala sam što više sažeti te istaknuti najvažnije i najzanimljivije sadržaje.

3. Uvod u baze podataka i sustave za upravljanje bazama podataka

U današnje vrijeme gotovo je nemoguće izgraditi aplikaciju koja ne koristi baze podataka za pohranu i čuvanje podataka. Stoga je veliki izazov pred sustavima za upravljanje bazama podataka, budući da se podaci i njihova struktura stalno mijenjaju, a sve je veća potreba za pohranom velike količine podataka. Osim za čuvanje podataka, baze podataka su važne i za manipulaciju podacima. Dakle, nije više od ključne važnosti samo pohraniti podatke i čuvati ih, nego i moći njima manipulirati, mijenjati ih iz oblika u oblik i pohranjivati u onom obliku u kojemu su nam potrebni. Osim toga, važno je organizirati same podatke unutar baze podataka. Baze podataka tako dobivaju novi dinamički oblik.

Iz svega toga dolazi i sama definicija baze podataka koja kaže da je baza podataka kolekcija podataka, ograničenja i operacija koja reprezentira neke aspekte realnog svijeta. (Maleković, Rabuzin, 2016)

Međutim, kada kažemo baza podataka tada ne mislimo nužno na softver, nego samo na kolekciju podataka, odnosno zbirku zapisa, dok je softver zapravo sustav za upravljanje bazama podataka (kraće SUBP). Dakle, SUBP je programska podrška za izvođenje operacija nad bazom podataka. Za svaku bazu podataka postoji strukturni opis činjenica sadržanih u toj bazi koji se naziva shema. Sustavi za upravljanje bazama podataka mogu se podijeliti prema modelu podataka koji podržavaju. Tako se razlikuju hijerarhijski, mrežni, relacijski, temporalni, nerelacijski itd. Od svih njih najzastupljenije su relacijske baze podataka. (Wikipedia, 2019)

Osnovne karakteristike, odnosno zadaće koje mora obavljati svaki sustav za upravljanje bazama podataka su zaštita baze podataka od neovlaštenog korištenja, čuvanje integriteta baze podataka, omogućavanje obnavljanja sadržaja baze podataka u slučaju njegova gubitka, omogućavanje više korisnika pristup istim podacima baze podataka i to istovremeno (višekorisnički pristup), opis i rukovanje podacima i identificiranje optimalne strukture za najprikladnije upravljanje podacima. (Carić, Buntić, 2015)

Budući da je sustav za upravljanje bazama podataka MariaDB rješenje koje je između relacijskog i nerelacijskog pristupa, ukratko ću pojasniti osnovne karakteristike oba ova pristupa.

3.1. Relacijske baze podataka

Relacijske baze podataka temelje se na relacijskom podatkovnom modelu. Taj model razvio je matematičar Edgar Frank Codd. Osnovna karakteristika ovog podatkovnog modela je da se podaci reprezentiraju relacijama, odnosno tablicama te da svaki podatak mora biti dostupan preko neke kombinacije imena tablica ili atributa.

Osim ove dvije osnovne karakteristike Codd je definirao 13 pravila (12 pravila i nulto pravilo) kako bi SUBP bio relacijski (Carić, Buntić, 2015):

- *engl. The information rule* – odnosi se na predstavljanje informacija tj. podaci se jednostavno i dosljedno reprezentiraju na jedinstven način kao vrijednosti u relacijama (tablicama),
- *engl. The guaranteed access rule* – odnosi se na pravilo pristupa (obavezna dostupnost) tj. svaki podatak u tablici mora biti logički dostupan preko kombinacije imena tablica, vrijednosti primarnoga ključa i imena atributa,
- *engl. Systematic treatment of null values* – odnosi se na tretiranje nepoznatih vrijednosti (NULL vrijednosti). Vrijednost NULL se tretira kao nepoznata neovisno o tipu podataka. Nepoznata vrijednost nije isto što i prazan znak ili broj 0 (nula),
- *engl. Dynamic online catalog based on the relational model* – odnosi se na dinamički online katalog tj. na rječnik baze podataka s informacijama o relacijskoj shemi. Taj rječnik mora biti dostupan i pohranjen kao svi podaci u bazi. Nad tim podacima autorizirani korisnici mogu postavljati upite koristeći SQL upitni jezik,
- *engl. The comprehensive data sublanguage rule* – odnosi se na pravilo sveobuhvatnog jezika tj. mora postojati jezik za komunikaciju sa bazom podataka koji podržava relacijske operatore koji se odnose na modifikaciju podataka, definiciju podataka i administraciju,
- *engl. The view updating rule* – odnosi se na pravilo pogleda tj. omogućava definiranje tzv. tablice pogleda koja se sastoji od jedne SELECT naredbe koja dohvaća podatke iz jedne ili više tablica. Sve kreirane poglede sustav mora moći ažurirati,
- *engl. High-level insert, update and delete* – odnosi se na pravila ažuriranja skupova (visoka razina unosa, izmjene i brisanja) tj. podaci mogu biti preuzeti u skupovima podataka iz jedne ili više tablica, a operacije umetanja, ažuriranja i brisanja moraju biti podržane za skupove podataka, a ne samo za jedan redak jedne tablice,
- *engl. Physical data independence* – odnosi se na fizičku nezavisnost tj. aplikacije koje pristupaju podacima ne smiju biti ovisne o promjenama u fizičkom načinu spremanja podataka, odnosno, aplikacije koje pristupaju podacima neovisne su o metodi pristupa podacima i o strukturi i načinu spremanja podataka na medije,

- *engl. Logical data independence* – odnosi se na logičku nezavisnost tj. odnosi među tablicama se mogu mijenjati, a da se istovremeno ne utječe na funkcije aplikacija koje se spajaju na tablice. Promjena strukture baze podataka ne smije uzrokovati ponovnu izradu baze podataka ili aplikacije,
- *engl. Integrity independence* – odnosi se na nezavisnost integriteta podataka tj. SUBP se mora brinuti o integritetu podataka, a ne aplikacija izvana,
- *engl. Distribution independence* – odnosi se na distribuiranu nezavisnost tj. aplikacija mora nastaviti raditi i kada se uvede distribuirana verzija SUBP-a, ali i kada se distribuirana verzija centralizira,
- *engl. The nonsubversion rule* – odnosi se na pravilo o nenarušavanju tj. integritet podataka ne smije biti narušen zaobilaženjem pravila integriteta i ograničenja,
- *engl. Rule 0 (The foundation rule)* – odnosi se na nulto pravilo tj. da bi sustav za upravljanje bazama podataka bio relacijski mora koristiti isključivo relacije.

Najpoznatiji relacijski sustavi za upravljanje bazama podataka u današnjemu svijetu su MySQL, Oracle, Microsoft Access, MS SQL Server, PostgreSQL i mnogi drugi.

3.2. Nerelacijske baze podataka

Osnovna karakteristika nerelacijskih baza podataka je da takve baze nemaju strogo određenu shemu. One kao što sam naziv kaže nisu relacijske pa ne koriste relacijski model niti SQL jezik.

Prema modelu podataka nerelacijske baze podataka mogu se podijeliti na četiri kategorije:

- model ključ-vrijednost (vrijednostima se pristupa preko ključa),
- dokumentno orijentirane baze podataka (*engl. document-oriented database*) (pohrana hijerarhijskih struktura direktno u bazu, ovo su polu-strukturirane baze podataka implementirane bez tablica),
- model stupca (koristi red i stupac kao ključeve),
- grafovske baze (u obliku grafova se prikazuju entiteti i veze između entiteta).

U vezi s nerelacijskim bazama podataka često se koristi pojam CAP teorem koji podrazumijeva 3 svojstva: konzistentnost, dostupnost i tolerancija particija. Konzistentnost se odnosi na to da svi čvorovi imaju identične podatke dostupne za transakcije. Dostupnost znači da će se svaki zahtjev za podatke izvršiti uspješno ili će se javiti poruka kako se zahtjev ne može izvršiti. Tolerancija particija znači da će sustav nastaviti svoj rad i u slučaju da se prekine veza među čvorovima. (Tivanovac, 2016)

4. Uvod u SUBP MariaDB

Sustav za upravljanje bazama podataka MariaDB donio je nove ideje u svijet baza podataka te uspio spojiti elemente iz relacijskog i nerelacijskog modela iz čega su proizašle i njegove glavne karakteristike te popularnost. U ovome poglavlju će biti riječ o nastanku ovog sustava, biti će iznijet osnovni pregled njegovih karakteristika, pregled podsustava koje koristi te grubi pregled naredbi, tipova podataka itd. koje će kasnije biti prikazane na primjeru baze podataka u praktičnome dijelu rada.

4.1. Povijest razvoja SUBP MariaDB

Sustav za upravljanje bazama podataka MariaDB kreiran je od istih autora kao što je to i MySQL. Voditelj projekta za razvoj SUBP MariaDB je Michael Widenius koji osim što je osnivač MySQL-a je osnivač i Monty Program AB-a. SUBP MariaDB je, uostalom, i ime dobio po njegovoj mlađoj kćeri - Maria. Osim po svojim osnovnim karakteristikama i načinu rada MySQL i MariaDB dijele sličnosti i po imenu, budući da je i MySQL dobio naziv prema Widenuius-ovoj kćeri My.

SUBP MariaDB svijetu je predstavljen, ne tako davno, 29. listopada 2009. godine. Stoga, za njega možemo reći i da je nov sustav u svijetu baza podataka. Poticaj za razvoj ovoga sustava došao je zbog prodaje MySQL-a u vlasništvo korporacije Oracle. Naime, autori su željeli stvoriti nov sustav koji će biti open source, odnosno otvorenoga koda i dostupan svima na korištenje, a podupirati jednake funkcionalnosti kao i MySQL. Ideja je zadržati što veću kompatibilnost s MySQL-om kako bi prijelaz s jednog sustava na drugi bio što jednostavniji. Prva dostupna verzija MariaDB Servera je verzija 5.1 izdana prije 10 godina. Posljednja verzija je 10.4 izdana 9. studenog 2018. godine. (Wikipedia, 2019)

Danas se SUBP MariaDB može pohvaliti da je prvi najbrže rastući SUBP na svijetu u kategoriji nerelacijskih sustava za upravljanje bazama podataka. Prvo mjesto drži posljednje 3 godine. Osim toga, MariaDB se može pohvaliti i da ju koriste velike tvrtke kao što su ServiceNow, DBS Bank, Google, Mozilla te Wikimedia Foundation.

Sustav je pisan u programskim jezicima C, C++, Pearl i Bash, a originalno je izdan od strane MariaDB Corporation AB i MariaDB Foundation. Trenutno je dostupan za platforme Linux, Windows i macOS. Osim toga, sustav ima i GNU licencu, a izvorni kod je dostupan na github repozitoriju na poveznici: https://github.com/MariaDB/server (Wikipedia, 2019).

4.2. Instalacija paketa MariaDB Server

Instalacija paketa je vrlo jednostavna. MariaDB Server dostupan je za besplatno preuzimanje na službenim stranicama, odnosno preko poveznice: https://mariadb.com/downloads/.

Potrebno je samo odabrati verziju koju želimo te operacijski sustav na koji ćemo instalirati. Za potrebe ovoga rada ja sam preuzela verziju 10.3.14-GA budući da je to posljednja izdana stabilna verzija, te platformu MS Windows (x64). Nakon odabira opcija paket se preuzme te instalira bez problema kao i svi uobičajeni alati. Na početku je moguće odabrati što sve iz paketa želimo instalirati, te poslije tog se savjetuje postavljanje lozinke za korijenskog korisnika, no to nije obvezno. Nakon toga moguće je odabrati TCP port, u mom slučaju to je bio port 3306. Nakon toga instalacija je završena i alat je spreman za korištenje. Sa SUBP MariaDB moguće je raditi u uobičajenoj konzoli, ali i povezati s nekim od drugih alata. Za potrebe ovoga rada ja ću koristiti alat JetBrains DataGrip 2018.

4.3. Osnovne karakteristike i rad sa SUBP MariaDB

Kako su sustavi MariaDB i MySQL binarno kompatibilni vrlo su male razlike u njihovim karakteristikama, pa slobodno možemo reći da SUBP MariaDB dijeli neke karakteristike sustava MySQL. Oba sustava zapravo mogu izvršavati isti izvršni kod, koriste jednake nazive datoteka, jednake putanje, biblioteke za povezivanje s drugim programskim jezicima, jednake protokole te još mnogo toga.

Osnovne karakteristike kojima SUBP MariaDB proširuje funkcionalnosti MySQL-a jesu te što omogućava podršku za statističku obradu podataka, omogućava rad s geografskim podacima te sadrži različite funkcije za obradu teksta. Jedna od osnovnih prednosti SUBP MariaDB te time svakako i karakteristika jest sigurnost baze podataka. Ova sigurnost proizlazi najprije iz suradnje autora sustava s organizacijom Mitre koja se bavi otkrivanjem ranjivosti programskih rješenja. Sigurnosti doprinosi i to što je sustav open source odnosno izvorni kod je svima dostupan, a osim toga u radu sa sustavom se koristi i enkripcija.

Od verzije 10.1 uvedene su i nove karakteristike kao što su dinamički i virtualni stupci te rad s geometrijskim podacima. Također, od te verzije omogućen je rad, odnosno integracija s alatom Galera koji omogućava rad s klasterima te time SUBP MariaDB omogućava master-master replikaciju (MySQL podržava master-slave replikaciju). (Stefanović, 2018)

4.3.1. Podsustavi SUBP MariaDB

SUBP MariaDB radi na način da integrira nekoliko podsustava koji podržavaju rad s bazama podataka. Neki od tih podsustava su: XtraDB, InnoDB, TokuDB, MyISAM, Memory, CSV, OQGraph, SpinxSE, MROONGA, Cassandra… Ovi podsustavi koriste se za izvršavanje upita i vraćanja rezultata postavljenog upita. Osim izvršavanja upita, i čitanje i pisanje podataka se delegira podsustavima. Podsustavi mogu pružati podršku i u izvođenju transakcija, kreiranju indeksa te referencijalnoga integriteta, a sve ovisi o podsustavu koji je uključen i delegiran. Određeni podsustav uključuje se korištenjem klauzule ENGINE. Ukoliko ne uključimo niti jedan specifični podsustav koristi se defaultni, a to je sustav InnoDB. Ako želimo pogledati uključene podsustave to se dobiva izvršavanjem naredbe SHOW ENGINES.

```
MariaDB [(none)]> SHOW ENGINES;
+--------------------+---------+----------------------------------------------------------------------------------------------+--------------+------+------------+
| Engine             | Support | Comment                                                                                      | Transactions | XA   | Savepoints |
+--------------------+---------+----------------------------------------------------------------------------------------------+--------------+------+------------+
| CSV                | YES     | Stores tables as CSV files                                                                   | NO           | NO   | NO         |
| MRG_MyISAM         | YES     | Collection of identical MyISAM tables                                                        | NO           | NO   | NO         |
| MEMORY             | YES     | Hash based, stored in memory, useful for temporary tables                                    | NO           | NO   | NO         |
| MyISAM             | YES     | Non-transactional engine with good performance and small data footprint                      | NO           | NO   | NO         |
| SEQUENCE           | YES     | Generated tables filled with sequential values                                               | YES          | NO   | YES        |
| InnoDB             | DEFAULT | Supports transactions, row-level locking, foreign keys and encryption for tables             | YES          | YES  | YES        |
| Aria               | YES     | Crash-safe tables with MyISAM heritage                                                       | NO           | NO   | NO         |
| PERFORMANCE_SCHEMA | YES     | Performance Schema                                                                           | NO           | NO   | NO         |
+--------------------+---------+----------------------------------------------------------------------------------------------+--------------+------+------------+
8 rows in set (0.033 sec)
```

Slika 1: Rezultat naredbe SHOW ENGINES

Izvršavanjem naredbe SHOW ENGINES na tek instaliranom SUBP MariaDB dobiva se popis prikazan na Slici 1. Iz dobivene tablice je vidljivo koje podsustave podržava trenutno instalirana verzija te koje funkcionalnosti su podržane određenim podsustavom. Također je vidljivo da je aktivan defaultni podsustav InnoDB. Taj podsustav podržava transakcije, čuvanje među stanja prilikom izvršavanja transakcije, XA1 transakcije te rad sa vanjskim ključevima. Zbog svojih performansi on je najkorišteniji podsustav.

4.3.2. Upitni jezici

SUBP MariaDB koncept je koji spaja karakteristike relacijskih i nerelacijskih baza podataka pa tako za rad s karakteristikama relacijskih baza podataka koristi SQL upitni jezik, dok za rad s karakteristikama nerelacijskih baza podataka upotrebljava koncepte NoSQL baza podataka. Neke od NoSQL naredbi koje su prisutne u SUBP MariaDB biti će opisane u nastavku.

4.3.2.1. Osnovne NoSQL naredbe

Za postavljanje upita nad bazom podataka moguće je koristiti NoSQL naredbu HANDLER koja daje direktan pristup za čitanje podataka iz baze, što je puno brže nego uobičajeni postupak sa SELECT naredbom jer nije potreban optimizator upita. Tako naredba HANDLER OPEN otvara tablicu, a HANDLER READ čita podatke. Nakon ključne riječi HANDLER definira se naziv tablice iz koje se želi čitati. Uz naredbu HANDLER moguće je kombinirati READ FIRST, READ NEXT, LIMIT, WHERE, CLOSE itd.

Za čitanje podataka iz neke datoteke koristi se NoSQL naredba LOAD_FILE. Ova naredba čita datoteku i vraća njen sadržaj u obliku stringa. Potrebno je navesti putanju do datoteke u oblim zagradama i imati ovlasti za čitanje navedene datoteke. (NoSQL, MariaDB, 2019)

Od ostalih karakteristika NoSQL jezika MariaDB podržava dinamičke stupce o kojima će biti riječ nešto kasnije.

4.3.2.2. Osnovne SQL naredbe

Za rad s relacijskim karakteristikama, kako je već navedeno, koristi se SQL upitni jezik. Tako za kreiranje baze podataka koristi se naredba CREATE DATABASE u kombinaciji sa željenim imenom baze podataka. Za rad nad određenom bazom podataka koristi se naredba USE u kombinaciji s nazivom željene baze podataka. Za brisanje tu je naredba DROP DATABASE.

Korištenjem SQL-a moguće je kreirati korisnika i ulogu i to naredbama CREATE USER za kreiranje korisnika te CREATE ROLE za kreiranje uloge. Osim kreiranja moguće je brisati te ažurirati korisnike ili uloge naredbama DROP za brisanje i ALTER za uređivanje. Za dodjeljivanje i uskraćivanje određenih prava ulogama koriste se uobičajene naredbe GRANT i REVOKE.

Za rad s tablicama, također, se koriste naredbe CREATE, DROP i ALTER u kombinaciji s ključnom riječi TABLE. Manipulacija podacima u tablicama može se postići naredbama: INSERT za umetanje, UPDATE za ažuriranje te DELETE za brisanje podataka. Kako bi mogli pregledavati kreirane tablice, njihova svojstva, kreirane poglede, okidače i dr., MariaDB koristi naredbu SHOW u kombinaciji s ključnom riječi TABLE za ispis tablica, CREATE TABLE za ispis svojstava tablice, odnosno ispis definirane strukture određene tablice, SHOW TRIGGERS ispisuje kreirane okidače itd.

Kako bi čitanje podataka iz tablica bilo moguće tu je naredba SELECT koja se koristi za postavljanje upita. Nju je moguće kombinirati s ključnim riječima COUNT, GROUP BY,

ORDER BY, LIMIT i dr. Sve ove naredbe i njihovo korištenje biti će prikazane u praktičnom dijelu rada na primjerima. (SQL Statemensts, MariaDB, 2019)

4.3.3. Tipovi podataka

Kod tipova podataka, koje je moguće koristiti u radu sa SUBP MariaDB, razlikuju se numerički tipovi, tekstualni tipovi, tipovi za rad s datumom i vremenom i drugi tipovi.

Od numeričkih tipova podataka moguće je koristiti: TINYINT, SMALLINT, MEDIUMINT, INT, BIGINT za cjelobrojne podatke, BOOLEAN ili TINYINT za logičke podatke tj. pohranu podataka u obliku 0 ili 1, odnosno istina ili laž (*engl. true/false*) te DECIMAL, FLOAT i DOUBLE za rad s decimalnim brojevima, odnosno brojevima s pomičnim zarezom (decimalni zarez ili decimalna točka). U kombinaciji s definicijom tipa podatka mogu se koristiti još i naredbe SIGNED i UNSIGNED za definiranje predznaka.

Od tekstualnih tipova podataka (*engl. string data types*) moguće je koristiti: CHAR, VARCHAR, BINARY, VARBINARY, BLOB, TINYBLOB, TEXT, LONGTEXT, TINYTEXT, MEDIUMTEXT i ENUM. Od ovih tipova u najčešćoj uporabi su VARCHAR i TEXT.

Za rad s datumom i vremenom koriste se sljedeći tipovi: DATE za definiranje datuma, TIME za definiranje vremena, DATETIME i TIMESTAMP za kombinaciju datuma i vremena.

Od ostalih tipova podataka postoje NULL vrijednosti te geometrijski tipovi podataka poput POINT, LINESTRING, POLYGON, MULTIPOINT, GEOMETRY itd. (Data Types, MariaDB, 2019)

4.4. Napredne karakteristike SUBP MariaDB

Od naprednijih karakteristika koje podržava sustav za upravljanje bazama podataka MariaDB u ovome potpoglavlju biti će pojašnjeno napredno pretraživanje teksta, rad s geografskim i geometrijskim podacima, transakcije i vanjski ključevi te virtualni i dinamički stupci.

4.4.1. Transakcije i vanjski ključevi

Kako je već rečeno SUBP MariaDB posjeduje karakteristike relacijskog i nerelacijskog modela podataka pa je kao takav specifičan sustav u svijetu baza podataka. Jedno od najbitnijih svojstava, odnosno karakteristika relacijskih baza podataka jesu transakcije. Transakcije kao takve SUBP MariaDB ne podržava, ali se one mogu omogućiti, također, uključivanjem određenih podsustava. Transakcije podržavaju InnoDB, TokuDB i XtraDB.

Rad sa stranim, tj. vanjskim ključevima, također, kao i transakcije nije direktno podržan u SUBP MariaDB, ali se može omogućiti njihovo korištenje uključivanjem odgovarajućeg podsustava koji podržava tu karakteristiku. Budući da je podsustav InnoDB automatski uključen prilikom instalacije paketa SUBP MariaDB, time je direktno i omogućen rad sa vanjskim ključevima. Naravno, ukoliko ne želimo raditi s podsustavom InnoDB vanjski ključevi omogućeni su i u nekim drugim podsustavima.

4.4.2. Napredno pretraživanje teksta

Defaultni podsustav InnoDB podržava funkcionalnost naprednog pretraživanja teksta (engl. *full-text search*), ali ta funkcionalnost može se koristiti i uključivanjem nekih drugih podsustava kao što su Aria, Mroonga itd. Ona omogućava pretraživanje po svim riječima svih dokumenata koji su pohranjeni u sustavu baza podataka. Postiže se korištenjem indeksa kreiranih nad tekstualnim stupcima. Takav indeks je tipa FULLTEXT te se može kreirati nad podacima koji su tipa CHAR, VARCHAR ili TEXT. Indeks se može kreirati u sklopu naredbe CREATE TABLE ili uobičajenom naredbom CREATE INDEX nad već postojećom tablicom. Osim tako moguće ga je kreirati i naredbom ALTER INDEX nad već postojećim indeksom na postojećoj tablici. (Stefanović, 2018)

Ovi indeksi, odnosno funkcionalnost naprednog pretraživanja teksta, omogućavaju da definiramo ukoliko želimo da se u rezultatu neke funkcije ili upita pojavljuje određena riječ ili da se ona ne pojavljuje. To je moguće učiniti korištenjem funkcije MATCH i AGAINST gdje se funkciji MATCH prosljeđuje naziv stupca nad kojim želimo pretražiti, a funkciji AGAINST parametri pretrage.

Napredno pretraživanje teksta odvija se na 3 načina:

1. Način prirodnoga jezika (engl. *natural langugage mode*) – ovaj način će izvršiti upit tako što će pretražiti podatke ili dokumente koji imaju barem jednu od navedenih riječi koje su proslijeđene kao parametri.

2. Način pretrage kombiniranjem operatora (engl. *boolean mode*) – ovaj način omogućava pisanje kompleksnih upita korištenjem operatora. Takav način omogućava da se iz pretrage isključe dokumenti ako sadrže određenu riječ, moguće je dati na važnosti određenoj riječi ili pretraživati na temelju neke fraze.

3. Način proširene pretrage (engl. *query expansion search*) – omogućava pretraživanje kroz 2 upita. Kod izvršavanja prvog upita koristi se način prirodnoga jezika, a nakon toga se u dobivenom rezultatu pretražuju najrelevantnije riječi.

4.4.3. Rad s geografskim i geometrijskim podacima

Kako bi omogućio rad s geografskim i geometrijskim podacima sustav za upravljanje bazama podataka MariaDB podržava tzv. stupce prostornoga tipa. Ovakve tipove moguće je koristiti uključivanjem podsustava MyISAM, InnoDB koji je defaultni sustav te uključivanjem podsustava Archive.

Osnovni tip podataka je tip GEOMETRY, ali za detaljan rad s geografskim i geometrijskim podacima potrebni su neki specifičniji tipovi podataka za koje možemo reći da su podtipovi tipa GEOMETRY to su primjerice: POINT, LINESTRING, POLYGON, MULTIPOINT, MULTILINESTRING, MULTIPOLYGON i GEOMETRYCOLLECTION. Kako je podatke ovoga tipa teško pretraživati uobičajenim mehanizmima pretraživanja, u svrhu rada s geografskim i geometrijskim podacima postoje prostorni indeksi (*engl. spatial index*). Ovakvi indeksi kreiraju se uobičajenim postupkom kao i svi normalni indeksi uz dodatak ključne riječi SPATIAL. Za prostorne indekse postoji jedna specifičnost u odnosu na ostale poznate indekse, a to je da stupci nad kojima se kreira prostorni indeks ne smiju biti prazni, odnosno ne smiju imati vrijednost NULL.

4.4.4. Virtualni stupci

Virtualni stupci jedna su od osnovnih karakteristika SUBP MariaDB koja ga razlikuje od ostalih sustava za upravljanje bazama podataka. Rad s virtualnim stupcima moguć je od verzije 5.2.

Virtualni stupci u literaturi se još nazivaju i generirani ili izračunati budući da se njihove vrijednosti dobivaju rješavanjem nekog deterministčkog izraza preko kojega se definiraju. Razlikuju se perzistentni i virtualni tip ovakvih stupaca. Kod perzistentnog tipa vrijednosti su sačuvane u tablici, dok se kod virtualnog vrijednosti svaki put ponovno izračunavaju prilikom upita nad tablicom. Podrazumijevani tip je virtualni. (Stefanović, 2018)

Tablica u kojoj se koristi mogućnost virtualnih stupaca kreira se jednakom naredbom kao i obične tablice, odnosno naredbom CREATE TABLE, razlika je ta što je pored definicije stupca za koji želimo da je virtualni potrebno definirati izraz po kojemu će se vrijednost tog stupca izračunavati te dodati ključnu riječ VIRTUAL za virtualni tip stupca ili PERSISTENT za perzistentni tip. Dakle, kod definiranja virtualnog stupca prvo se piše naziv stupca zatim tip podataka koji će se spremati, nakon toga u oblim zagradama definira se izraz po kome će se računati vrijednost i nakon toga slijedi ključna riječ VIRTUAL ili PERSISTENT.

Vrijednosti virtualnih ili perzistentnih stupaca ne mogu se mijenjati. Ukoliko, izričito želimo postaviti neku podrazumijevanu vrijednost moguće je koristiti NULL ili DEFAULT. Što

se definiranja izraza po kome će se izračunavati vrijednost stupca tiče tu valja slijediti sljedeća pravila:

- Nije dozvoljeno koristiti podizraze ili vrijednosti koje se ne nalaze u trenutnom redu.
- Nije moguće koristiti vrijednosti koje su definirane nakon trenutnog virtualnog stupca. Dakle, ukoliko želimo koristiti vrijednost nekoga stupca u izračunu taj stupac je potrebno definirati prije virtualnog kako bi se mogla preračunati vrijednost tekućeg stupca.

Indeks je moguće kreirati samo nad perzistentnim tipom virtualnih stupaca. Perzistetni tip može biti dio vanjskog ključa, ali ne može biti primarni ključ niti dio primarnog ključa. Virtualni tip ne može biti niti primarni ključ, niti dio primarnog ključa, a ne može biti ni vanjski ključ kao ni dio vanjskoga ključa. (Stefanović, 2018)

Virtualne stupce, kao i druge karakteristike, moguće je koristiti ukoliko je uključen odgovarajući podsustav. Ova karakteristika omogućena je korištenjem InnoDB, Aria, MyISAM i CONNECT podsustava.

4.4.5. Dinamički stupci

Kako su transakcije i njihovo korištenje osnovna karakteristika relacijskih sustava za upravljanje bazama podataka, tako su dinamički stupci karakteristika nerelacijskih sustava za upravljanje bazama podataka. Budući da je SUBP MariaDB između ta dva modela podataka, on podržava dinamičke stupce kao karakteristiku nerelacijskih sustava za upravljanje bazama podataka. Dinamičke stupce moguće je koristiti od verzije 5.3. Ova karakteristika omogućava da se skup različitih stupaca smjesti u okviru jednog stupca.

U bazi podataka dinamički stupci predstavljaju se kao podaci tipa TINYBLOB, BLOB, MEDIUMBLOB i LONGBLOB. Za rad s njima koriste se posebno definirani skupovi funkcija. Dinamički stupci svoju primjenu nalaze kada je potrebno smjestiti različite atribute koji nam nisu unaprijed poznati ili ukoliko podaci imaju različita svojstva. Kreiranje indeksa nad dinamičkim stupcima nije moguće. (Stefanović, 2018)

Tablica koja sadrži dinamičke stupce kreira se uobičajenom naredbom CREATE TABLE, zatim je potrebno kod stupca za koji želimo da bude dinamički tip podataka postaviti na neki od tipova podataka BLOB. Vrijednosti dinamičkih stupaca postavljaju se prilikom INSERT operacije korištenjem funkcije COLUMN_CREATE koja kao parametre prima nazive stupaca i njihove vrijednosti. Za čitanje se uz uobičajen SELECT kombinira funkcija COLUMN_GET. Za brisanje je to funkcija COLUMN_DELETE, kod ažuriranja se može dodati dodatan stupac funkcijom COLUMN_ADD. Za rad s dinamičkim stupcima još postoje funkcije COLUMN_EXISTS ukoliko želimo provjeriti postojanje stupca s proslijeđenim nazivom,

COLUMN_CHECK koristi se za provjeru sadržaja stupca i COLUMN_LIST vraća listu s nazivima svih stupaca koji su u važećem dinamičkom stupcu.

4.5. Prednosti i nedostaci SUBP MariaDB

Kao i svaki SUBP dostupan na tržištu, tako i SUBP MariaDB ima svoje prednosti i nedostatke. Glavne prednosti ogledaju se u glavnim karakteristikama ovoga sustava opisanim u prethodnom tekstu. Iz čega se da zaključiti da ovaj relativno mlad sustav ima više prednosti nego nedostataka u odnosu na mnogobrojne konkurente na tržištu. Sasvim je jasno da je glavna prednost to što je korištenje besplatno te dostupno svima, a osim toga što je korištenje omogućeno svima i sam izvorni kod je javno dostupan. Kako većina ostalih prednosti proizlazi iz već opisanih karakteristika u nastavku ću ih samo pobrojiti te ukratko pojasniti razloge zbog kojih se u literaturi upravo one navode kao prednosti:

- ✓ open source: izvorni kod je dostupan svima preko github platforme, korisnici mogu vidjeti samu implementaciju te im je omogućeno predlagati određene promjene za koje smatraju da su potrebne. Budući da je alat open source vrlo aktivno se mijenja i poboljšava kako u implementacijskom tako i u dokumentacijskom smislu,
- ✓ mogućnost rada s geometrijskim i geoprostornim podacima: omogućeno je spremanje koordinata i lako pretraživanje prostornih podataka,
- ✓ dinamički stupci: dozvoljavaju uporabu tzv. NoSQL funkcionalnosti pa tako jedna baza podataka može biti izrađena kombiniranjem SQL-a i NoSQL-a,
- ✓ mnogo podsustava s kojima se lako integrira: ovi podsustavi omogućavaju više mjesta za pohranu, rad s naprednijim funkcionalnostima, posebnim vrstama indeksa, a osim što nude brojne mogućnosti podsustavi dolaze u paketu sa službenom verzijom sustava pri čemu korisnik automatski ima garanciju da će SUBP MariaDB i odabrani podsustav iz paketa biti kompatibilni,
- ✓ bolje performanse: studije pokazuju da je SUBP MariaDB radikalno brži sustav od MySQL-a, što je omogućeno putem optimizatora upita koji omogućava brže izvršavanje postavljenih zadataka,
- ✓ lak prelazak s MySQL-a na MariaDB: ova dva sustava su potpuno kompatibilna pa je migracija s jednog na drugi vrlo jednostavna, potrebno je samo deinstalirati onaj koga više ne želimo koristiti te instalirati drugi.

Kao i svaki sustav tako i ovaj ima i svoje mane kojih je znatno manje nego prednosti. Kao osnovni nedostatak ovog SUBP-a navodi se to što je relativno mlad u odnosu na svoje glavne konkurente. Budući da je nov sustav na tržištu teško mu je pridobiti nove korisnike jer

većina velikih kompanija igra na sigurno te će radije odabrati neki već isprobani i ranije korišteni sustav bez mnogo improvizacije i isprobavanja novih stvari. (Cvetemirov Ivanov, 2019)

Što se tiče funkcijskih nedostataka, korisnici navode sljedeće (TrustRadius, 2019):

- slabije performanse na operacijskom sustavu macOS u odnosu na Windows,
- ukoliko je brzina ključan element za odabir SUBP-a, MariaDB je u dobroj poziciji, ali još uvijek postoje neke brže alternative poput Amazon Aurora,
- starije verzije MariaDB Servera nemaju dodatak za uporabu JSON-a, u novijim verzijama omogućen je taj dodatak, ali još uvijek nije potpuno jasno definirana njegova uporaba,
- sporije brisanje podataka iz predmemorije u odnosu na konkurente,
- nesigurna budućnost sustava, budući da je razvijen od strane male korporacije,
- mala količina dostupne literature za proučavanje sustava,
- nedostatak naredbe za računanje medijana,
- mal broj alata s kojima se moguće povezati za rad s bazom podataka uz pomoć grafičkoga sučelja,
- kompliciraniji prelazak s MariaDB na MySQL u odnosu na prijelaz s MySQL-a na MariaDB.

5. Usporedba SUBP MariaDB sa SUBP MySQL

Postojanje konkurencije nije samo poticajno u svijetu marketinga i velikih tvrtki nego i u svijetu informatike pa tako i baza podataka. Glavni konkurenti MariaDB i MySQL tako su potaknuli jedni druge na promjene i inovacije kako bi pridobili što više korisnika. U ovome poglavlju stoga će biti riječ o sličnostima ova dva sustava i njihovim zajedničkim karakteristikama, a osim toga naglasak će biti i na razlikama koje su ipak važnije u pridobivanju korisnika na svoju stranu.

5.1. Usporedba po osnovnim karakteristikama

Prvi kriterij po kome ću usporediti ova dva sustava jesu njihovi korisnici. Kako je MySQL mnogo stariji na tržištu sustava za upravljanje bazama podataka, gdje je prisutan od 1995. godine, za očekivati je da mu je korisnička baza nešto veća. Tako glavnu podršku tom sustavu daju velike tvrtke: GitHub, US Navy, NASA, Tesla, Netflix, WeChat, Facebook, Zendesk, Twitter, Zappos, YouTube, Spotify i mnoge druge poznate kompanije, a čitav popis moguće je pronaći na službenim stranicama MySQL-a (https://www.mysql.com/customers/). No na tom polju ne zaostaje mnogo ni MariaDB. Iako se na tržištu ovaj sustav pojavio 2009. godine danas se može pohvaliti da njegove usluge koriste Google, Craigslist, Wikipedia, Archlinux, RedHat, CentOS, Fedora i druge.

Osim što se oba sustava, dakle, mogu pohvaliti velikim imenima među suradnicima još jedna sličnost je u strukturi i sintaksi samih sustava. Budući da je sustav MariaDB razvijen od istih autora kao i MySQL to je i za očekivati. SUBP MariaDB razvijen je na temeljima MySQL-a tako da su naredbe i rad jednaki u oba sustava. Oba podržavaju relacijski model podataka, koriste tablice, ograničenja, okidače, uloge, pohranjene procedure, poglede i ostale karakteristike sustava za upravljanje bazama podataka. Ove zajedničke osobine omogućavaju lak prijelaz s jednoga sustava na drugi bez mnogo truda. (Sarig, 2019)

Dakle, pravo je pitanje u čemu je onda razlika između ova dva sustava. Temeljna razlika je u performansama. SUBP MariaDB je razvio nekoliko optimizacija kako bi poboljšao performanse u odnosu na MySQL što se pokazalo i u brojim testovima i istraživanjima. Bolje performanse bile su i razlog nastanka samoga sustava za upravljanje bazama podataka MariaDB. Nešto više o performansama u narednom potpoglavlju.

Kako bi usporedbu ova dva kompatibilna SUBP-a još bolje pokazala u nastavku slijedi ilustrativna tablica s još nekim zanimljivim podacima. (Izvor tablice: MySQLTUTORIAL, 2019 i Solid IT, 2019).

Tablica 1:Usporedba MariaDB i MySQL

	MySQL	MariaDB
Vlasnik	Oracle Corporation	MariaDB Corporation AB, MariaDB Foundation
Protokoli	MySQL	MySQL + MariaDB
Izvorni kod	Open Source + dio je u vlasništvu Oracle Corporation	Open Source
Razvoj	Zatvoreni	Otvoreni
Prva verzija	1995. godine	2009. godine
Operacijski sustavi	FreeBSD Linux OS X Solaris Windows	FreeBSD Linux Solaris Windows
Primarni model podataka	Relacijski	Relacijski
Memorijski podsustavi	InnoDB MyISAM BLACKHOLE CSV MEMORY ARCHIVE MERGE	InnoDB MyISAM BLACKHOLE CSV MEMORY ARCHIVE MERGE ColumnStore MyRocks Aria SphinxSE TokuDB CONNECT SEQUENCE Spider Cassandra

CHECK ograničenje	Nema	Ima
DEFAULT izrazi	Nema	Ima + podržava DEFAULT vrijednosti za podatke tipa BLOB i TEXT
Virtualni stupci	Ima	Ima
Dinamički stupci	Nema	Ima
Okidači	Ima	Ima
Vanjski ključevi	Ima	Ima
Uloge	Ima	Ima
DELETE … RETURNING	Ima	Ima
GIS podrška	Ima	Ima
Izvještaji o napretku za naredbe ALTER TABLE i LOAD DATA INFILE	Nema	Ima
SQL Managment	MySQL WorkBench	SQLyog
Backup	MySQL Enterprise Backup	MariaDB Backup
Podrška za JSON	Ima	Ima
Enkripcija	MySQL Enterprise Firewall	MariaDB Encryption

5.2. Usporedba po performansama

Kada je u pitanju usporedba ova dva sustava po performansama, dolazimo do značajnih razlika nego što je to bilo u usporedbi po osnovnim karakteristikama. Temeljne razlike slijede.

Prva razlika u performansama očituje se u korištenju pogleda (*engl. Database Views*). Pogledi su zapravo virtualne tablice u bazi podataka koje se mogu pretraživati jednako kao i sve ostale tablice u bazi. U MySQL-u upit nad pogledom zapravo povlači za sobom upit nad svim tablicama koje su povezane u pogled bez obzira na to jesu li podaci iz svih tablica

zahtijevani u glavnom upitu. Za razliku od tog pristupa, korištenjem SUBP MariaDB upit nad pogledom će pokrenuti upite samo nad tablicama čiji su podaci zatraženi u glavnom upitu. (Goel, 2019)

MariaDB svoje bolje performanse temelji i na boljoj memoriji korištenjem podsustava MyRocks, XtraDB, Aria te implementacijom Segmented Key Cache memorije i njenih svojstava. To poboljšanje omogućava lakši rad s više dretava. On omogućava da više dretvi može nesmetano raditi paralelno čime su poboljšane performanse baze podataka. (Goel, 2019)

Još jedna značajka koja ide u prilog SUBP MariaDB su virtualni stupci o kojima je već ranije bilo puno riječi. Ovu karakteristiku SUBP MySQL posjeduje, ali nije baš najsretnije izvedena, tako da je jasno da u tom segmentu zasigurno pobjeđuje MariaDB. Virtualni stupci omogućavaju izračunavanje raznih kalkulacija na razini baze podataka što je veoma korisno kada više aplikacija pristupa istom stupcu i potrebno je jednaku kalkulaciju zapisati u svaku aplikaciju.

Od verzije 10.0. SUBP MariaDB implementirao je još jednu karakteristiku koja poboljšava performanse. To je paralelno izvršavanje više upita koje nije moguće raditi u MySQL-u. (Shay, 2018)

6. Baza podataka za potrebe autoškole

Karakteristike odabranoga sustava za upravljanje bazama podataka MariaDB pokazat ću u ovome praktičnome dijelu rada na primjeru baze podataka za potrebe autoškole. Najprije ću pojasniti samu domenu baze podataka, prikazati pripadajući ERA model, a zatim krenuti sa samom implementacijom tablica, ključeva, ograničenja, okidača i dr.

6.1. Opis domene

Baza podataka za potrebe autoškole koju sam zamislila treba omogućavati unos, izmjenu, brisanje i pregledavanje popisa instruktora autoškole. Svaki instruktor može pripadati jednom tipu instruktora, odnosno on može biti ili predavač propisa o sigurnosti prometa ili instruktor vožnje, stoga korisniku nije omogućen unos, izmjena i brisanje podataka o tipu instruktora. Budući da svaka autoškola ima svoja vozila, potrebno je voditi evidenciju o vozilima koja su u posjedu, te o instruktorima koji imaju pristup tim vozilima. U ovoj bazi važna stavka jesu kategorije za koje je omogućena obuka u autoškoli, stoga je važno zabilježiti koji instruktor posjeduje kvalifikacije za obuku koje kategorije te koje vozilo je namijenjeno kojoj kategoriji. Baza podataka mora omogućavati vođenje evidencije o kandidatima koji pohađaju autoškolu, stoga je za svakoga kandidata potrebno bilježiti koju kategoriju po kojemu programu pohađa, voditi evidenciju o odslušanim satima obuke te na kraju voditi evidenciju o uspjehu kandidata na ispitu.

Kao i svaka druga baza podataka koja je rađena po stvarnome sustavu, tako i ova ima svoja pravila, stoga će biti potrebno ograničiti pojedine radnje nad bazom, što će biti detaljno opisano u narednim poglavljima.

Za ovu bazu najvažniji entiteti jesu „Instruktor“ (sadrži zapise o trenutnim instruktorima i njihovim kvalifikacijama koji su zaposleni u autoškoli), „Kandidat“ (sadrži evidenciju kandidata koji su pohađali ili pohađaju autoškolu), „Nastavni sat“ (sadrži evidenciju obavljenih sati obuke za svakoga kandidata, kategoriju, program i datum) te „Uspjeh kandidata“ (sadrži zapise o izlasku kandidata na ispit iz pohađanog programa te njegov uspjeh na istom).

6.2. Relacijska shema i ERA model

U dolje prikazanoj relacijskoj shemi primarni ključevi označeni su podebljanjem (bold), a vanjski ključevi su ukošeni (italic).

Tip_instruktora (**ID**, Naziv, Opis);

Instruktor (**ID**, Ime, Prezime, OIB, Email, Broj_telefona, *tip_id*);

Kategorija (**ID**, Oznaka, Opis);

Vozilo (**ID**, Vrsta, Registarske_oznake, Opis, *kategorija_id*);

Instruktor_vozilo (***insktuktor_id, vozilo_id***);

Instruktor_kategorija (***insktuktor_id, kategorija_id***);

Program (**ID**, Naziv, Broj_sati, Opis);

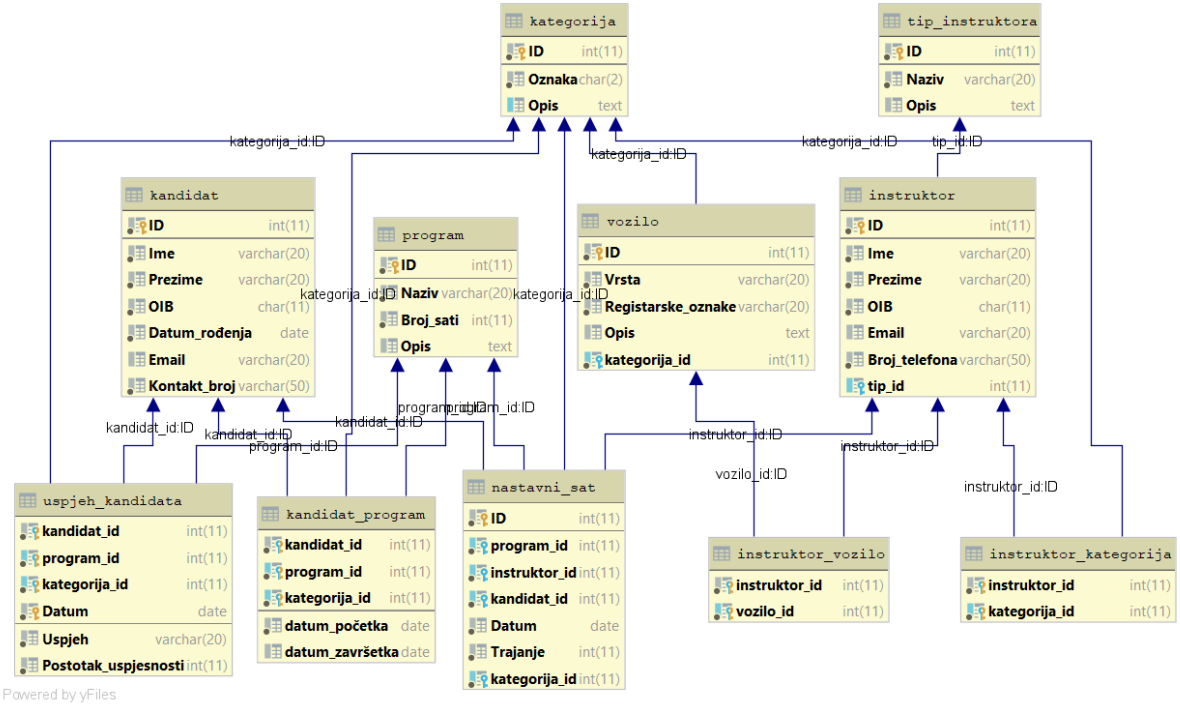
Kandidat (**ID**, Ime, Prezime, OIB, Datum_rođenja, Email, Kontakt_broj);

Kandidat_program (***kandidat_id, program_id, kategorija_id***, datum_početka, datum_završetka);

Nastavni_sat (**ID**, *program_id, instruktor_id, kandidat_id,* Datum, Trajanje, *kategorija_id*);

Uspjeh_kandidata (***kandidat_id, program_id, kategorija_id,*** **datum**, uspjeh, postotak_uspješnosti);

ERA model opisane baze podataka izrađen je u alatu JetBrains DataGrip 2018, a struktura mu je određena prethodno navedenome relacijskom shemom. Prikaz modela je na sljedećoj slici:



Slika 2: ERA model

6.3. Opis entiteta i atributa

U ovome sustavu jaki entiteti su Kategorija, Vozilo, Instruktor, Tip_instruktora, Kandidat, Nastavni_sat, Program i Uspjeh_kandidata. U tablicama koje slijede bit će opisani entiteti, njihovi atributi te po jedan primjer svake instance entiteta.

Tablica 2: Tip_instruktora

Tip_instruktora

Opis entiteta	Svaka autoškola ima svoje vrste instruktora koji mogu biti ili predavači za program propisa ili instruktori vožnje, stoga je važno zabilježiti sve te tipove u tablici kako bi kasnije instruktoru mogli dodijeliti njegov tip.
Atributi	ID – integer (autoincrement) primarni ključ relacije, služi za identifikaciju tipa instruktora, Naziv – varchar(20), naziv tipa instruktora, obavezno polje, Opis – text, dodatan opis tipa instruktora
Primjer	(1,'Predavač','Instruktor za održavanje satova propisa o sigurnosti prometa')

Tablica 3: Instruktor

Instruktor

Opis entiteta	Instruktori su ključni zaposlenici autoškole koji obučavaju kandidate stoga je važno njih bilježiti u posebnu tablicu koja sadrži osnovne informacije potrebne za bazu podataka.
Atributi	ID – integer (autoincrement) primarni ključ relacije, služi za identifikaciju instruktora, Ime – varchar(20), ime instruktora, obavezno polje, Prezime – varchar(20), prezime instruktora, obavezno polje, OIB – char(11), OIB instruktora, obavezno polje, E-mail – varchar(20), e-mail adresa instruktora Broj_telefona – varchar(50), kontakt broj instruktora, obavezno polje

	tip_id – integer, vanjski ključ na tablicu Tip_instruktora, identifikacija tipa instruktora kojemu instruktor pripada, obavezno polje.
Primjer	(1,'Pero','Perić','03265478945','pperic@gmail.com',0635897412,1)

Tablica 4: Kategorija

Kategorija

Opis entiteta	Vrste kategorija koje se obučavaju u autoškoli.
Atributi	ID – integer (autoincrement) primarni ključ relacije, služi za identifikaciju kategorije, Oznaka – char(2), oznaka kategorije, obavezno polje, Opis – text, dodatan opis kategorije
Primjer	(1,'A','motocikli sa ili bez bočne prikolice, motorna vozila na tri kotača čija je snaga veća od 15kw')

Tablica 5: Vozilo

Vozilo

Opis entiteta	Vozila u posjedu autoškole koja se koriste za obuku kandidata.
Atributi	ID – integer (autoincrement) primarni ključ relacije, služi za identifikaciju vozila, Vrsta – varchar(20), vrsta vozila, obavezno polje, Registarske_oznake – varchar(20), registarske oznake vozila, obavezno polje, Opis - text, dodatan opis vozila, kategorija_id – integer, vanjski ključ na tablicu Kategorija, označava kategoriju kojoj vozilo pripada i koja se može obučavati na vozilu, obavezno polje.
Primjer	(1,'automobil','VŽ1254','vozilo za poučavanje kategorije B',2)

Tablica 6: Instruktor_vozilo

Instruktor_vozilo

Opis entiteta	Sadrži podatke o tome koji instruktor posjeduje licencu za koju kategoriju vozila, odnosno u tablicu se zapisuje koji instruktor obavlja obuku na kojemu vozilu.
Atributi	instruktor_id – integer, dio primarnog ključa relacije, služi za identifikaciju instruktora, vanjski ključ na tablicu Instruktor, obavezno polje, vozilo_id – integer, dio primarnog ključa relacije, služi za identifikaciju vozila, vanjski ključ na tablicu Vozilo, obavezno polje.
Primjer	(1,1)

Tablica 7: Instruktor_kategorija

Instruktor_kategorija

Opis entiteta	Sadrži podatke o tome koji instruktor posjeduje licencu za koju kategoriju vozila, odnosno u tablicu se zapisuje koji instruktor ima pravo obučavati koju kategoriju.
Atributi	instruktor_id – integer, dio primarnog ključa relacije, služi za identifikaciju instruktora, vanjski ključ na tablicu Instruktor, obavezno polje, kategorija_id – integer, dio primarnog ključa relacije, služi za identifikaciju kategorije, vanjski ključ na tablicu Kategorija, obavezno polje.
Primjer	(1,1)

Tablica 8: Program

Program

Opis entiteta	Sadrži podatke o programima koji se mogu pohađati u autoškoli i broju sati nastave za taj program, u ovome slučaju postoje dva programa, a to su program propisa o sigurnosti prometa te program upravljanja motornim vozilom.

Atributi	ID – integer (autoincrement) primarni ključ relacije, služi za identifikaciju programa, Naziv – varchar(20), imenuje program, obavezno polje, Broj_sati – integer, broj sati obuke za program, obavezno polje, Opis – text, dodatan opis programa
Primjer	(1,'poznavanje propisa',30,' poznavanje propisa o sigurnosti prometa')

Tablica 9: Kandidat

Kandidat

Opis entiteta	Sadrži podatke o kandidatima koji su upisani u autoškolu.
Atributi	ID – integer (autoincrement) primarni ključ relacije, služi za identifikaciju kandidata, Ime – varchar(20), ime kandidata, obavezno polje, Prezime – varchar(20), prezime kandidata, obavezno polje, OIB – char(11), OIB kandidata, obavezno polje, Datum_rođenja – date, datum rođenja kandidata, obavezno polje, Email – varchar(20), e-mail adresa kandidata Kontakt_broj – varchar(50), kontakt broj kandidata, obavezno polje
Primjer	(1, 'Ivo', 'Ivić', '03258568945', '1991-11-22', 'iivic@gmail.com', 0635258412)

Tablica 10: Kandidat_program

Kandidat_program

Opis entiteta	Sadrži podatke o tome koji kandidat je upisao koji program za koju kategoriju, te kojega datuma je započeo s programom odnosno kojega datuma je završio program.
Atributi	kandidat_id – integer, dio primarnog ključa relacije, služi za identifikaciju kandidata, vanjski ključ na tablicu Kandidat, obavezno polje,

	program_id – integer, dio primarnog ključa relacije, služi za identifikaciju programa koji se upisuje, vanjski ključ na tablicu Program, obavezno polje, kategorija_id – integer, dio primarnoga ključa relacije, služi za identifikaciju kategorije koja se upisuje, vanjski ključ na tablicu Kategorija, obavezno polje, datum_početka – date, datum upisa programa, odnosno početka obuke programa za određenu kategoriju, obavezno polje, datum_završetka – date, datum završetka programa, odnosno polaganja programa za određenu kategoriju, unosi se nakon položenog ispita.
Primjer	(1,1,1,'2018-10-1',null)

Tablica 11: Nastavni_sat

Nastavni_sat

Opis entiteta	Sadrži podatke o prisutnosti kandidata na nastavnim satima, odnosno evidentira broj sati obuke koje je kandidat prošao za određenu kategoriju po određenom programu.
Atributi	ID – integer (autoincrement), identificira nastavni sat, primarni ključ relacije, program_id – integer, služi za identifikaciju programa koji se upisuje, vanjski ključ na tablicu Program, obavezno polje, instruktor_id – integer, služi za identifikaciju instruktora koji održava nastavni sat, vanjski ključ na tablicu Instruktor, obavezno polje, kandidat_id – integer, služi za identifikaciju kandidata koji pohađa nastavni sat, vanjski ključ na tablicu Kandidat, obavezno polje, Datum – date, datum održavanja nastavnog sata, obavezno polje, Trajanje – integer, trajanje nastavnog sata, obavezno polje, kategorija_id – integer, služi za identifikaciju kategorije za koju se sat održava, vanjski ključ na tablicu Kategorija, obavezno polje.
Primjer	(18,3,1,7,'2018-10-06',2,1)

Tablica 12: Uspjeh_kandidata

Uspjeh_kandidata

Opis entiteta	Sadrži podatke o uspjehu kandidata na ispitu iz upisanoga programa za upisanu kategoriju.
Atributi	kandidat_id – integer, služi za identifikaciju kandidata koji polaže ispit, dio primarnoga ključa relacije, vanjski ključ na tablicu Kandidat, obavezno polje, program_id – integer, služi za identifikaciju programa koji se polaže, dio primarnoga ključa relacije, vanjski ključ na tablicu Program, obavezno polje, kategorija_id – integer, služi za identifikaciju kategorije za koju se polaže ispit, dio primarnoga ključa relacije, vanjski ključ na tablicu Kategorija, obavezno polje, Datum – date, datum održavanja ispita, dio primarnoga ključa relacije, obavezno polje, Uspjeh – varchar(20), uspjeh kandidata na ispitu, obavezno polje, Postotak_uspješnosti – integer, postotak riješenosti ispita, obavezno polje.
Primjer	(7,3,1,'2018-11-05','nije polozio',20)

6.4. Kreiranje baze podataka

Za kreiranje baze podataka, u ovome slučaju baze podataka za potrebe autoškole, koristi se ključna riječ CREATE DATABASE. Dakle, bazu podataka kreirala sam izvršavanjem naredbe:

```
CREATE DATABASE autoskola;
```

Nakon kreiranja baze podataka, treba se spojiti na kreiranu bazu kako bi mogli u njoj kreirati željene tablice, okidače, poglede i slično to se može učiniti izvršavanjem naredbe:

```
USE autoskola;
```

U kreiranoj bazi pod nazivom autoskola kreirat ću novu ulogu i nazvati ju admin:

```
CREATE ROLE admin;
```

Kako bi se prebacila na novokreiranu ulogu izvršava se naredba:

```
SET ROLE admin;
```

Sada kada je kreirana potrebna baza i uloga, moguće je prijeći na sljedeći korak, a to je kreiranje tablica koje su prethodno definirane relacijskom shemom i pojašnjene u domeni. Prva tablica koja će biti kreirana je tablica Tip_instruktora budući da tablica Instruktor sadrži vanjski ključ na tip. Kreiranje tablice provodi se jednako kao i u MySQL-u izvršavanjem naredbe CREATE TABLE. Kod naredbe za kreiranje tablice Tip_instruktora slijedi:

```
CREATE TABLE Tip_instruktora (ID INT AUTO_INCREMENT PRIMARY KEY,
Naziv VARCHAR(20) NOT NULL,
Opis TEXT);
```

Za prikaz svih kreiranih tablica u trenutnoj bazi podataka koristi se naredba:

```
SHOW TABLES;
```

Nakon što je tablica kreirana moguće ju je popuniti podacima:

```
INSERT INTO Tip_instruktora (Naziv, Opis)
VALUES ("Predavac", "Instruktor za održavanje satova propisa o
sigurnosti prometa"),
("Instruktor vožnje", "Instruktor za održavanje satova vožnje");
```

Za pregledavanje umetnutih redaka koristi se:

```
SELECT * FROM tip_instruktora;
```

```
MariaDB [autoskola]> SELECT * FROM tip_instruktora;
+----+------------------+-----------------------------------------------------------+
| ID | Naziv            | Opis                                                      |
+----+------------------+-----------------------------------------------------------+
|  1 | Predavac         | Instruktor za održavanje satova propisa o sigurnosti prometa |
|  2 | Instruktor vožnje | Instruktor za održavanje satova vožnje                    |
+----+------------------+-----------------------------------------------------------+
2 rows in set (0.001 sec)
```

Slika 3: Ispis podataka iz tablice

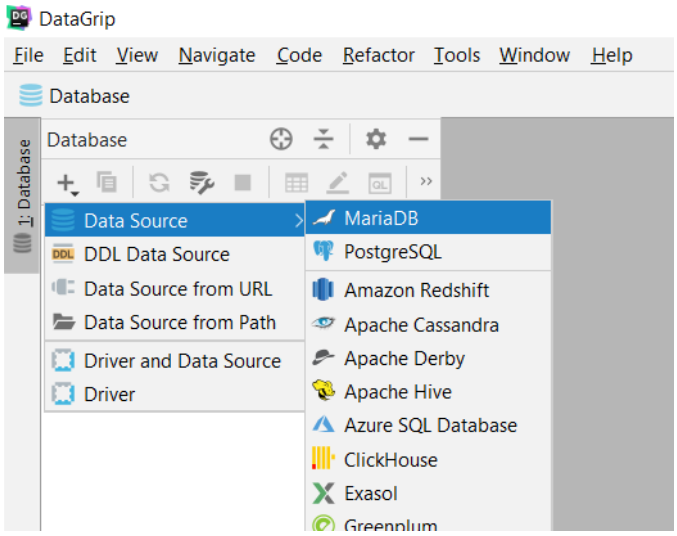
Kreiranje tablice koja u sebi sadrži vanjski ključ prikazati ću na primjeru tablice Instruktor:

```
CREATE TABLE Instruktor (ID INT AUTO_INCREMENT PRIMARY KEY,
Ime VARCHAR(20) NOT NULL,
Prezime VARCHAR(20) NOT NULL,
OIB CHAR(11) NOT NULL, Email VARCHAR(20),
```

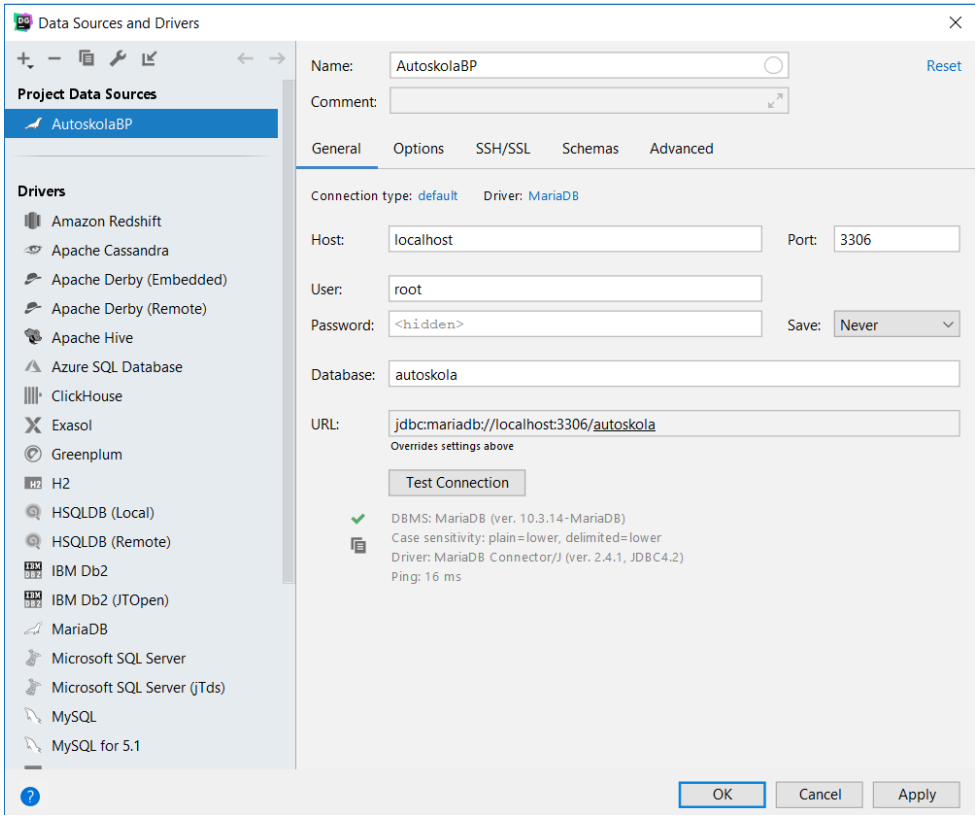
```
Broj_telefona VARCHAR(50) NOT NULL,
tip_id INT REFERENCES Tip_instruktora (ID) ON DELETE SET NULL ON UPDATE
CASCADE);
```

Uređivanje strukture kreiranih tablice moguće je korištenjem naredbe ALTER TABLE koja, poput i ostalih naredbi, ima jednaku sintaksu kao i u MySQL-u. Za brisanje koristi se DROP TABLE.

Ostale tablice kreirala sam pomoću alata JetBrains DataGrip 2018. No, prije kreiranja potrebno je spojiti se na bazu podataka. Potrebno je navesti Data Source što je u ovom slučaju MariaDB, zatim je potrebno upisati parametre potrebne za spajanje na bazu i testirati konekciju, ukoliko su svi podaci uneseni ispravno alat će se spojiti na odabranu bazu podataka pa je kreiranje tablica i rad s bazom omogućen. Kako smo se uspješno spojili na bazu podataka autoškola vidljive su tablice koje su prethodno kreirane putem konzole, dakle tablice tip_instruktora i instruktor. (Slike 4 i 5)

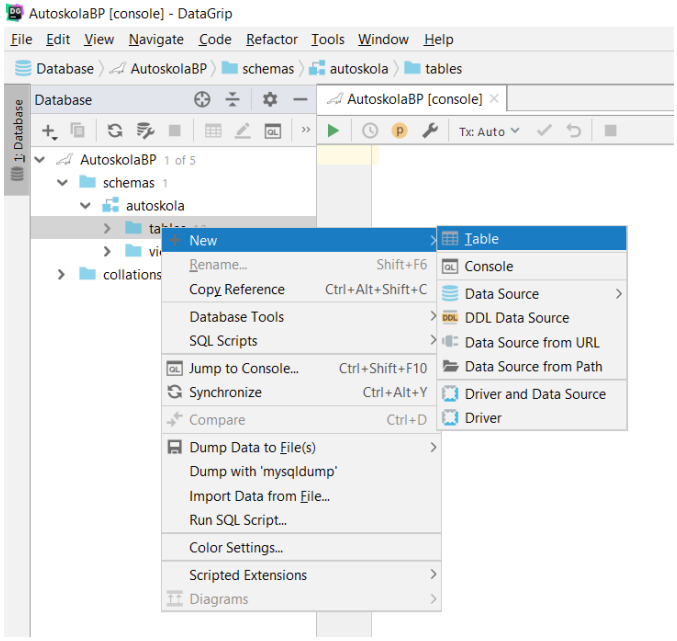


Slika 4: Spajanje na bazu podataka

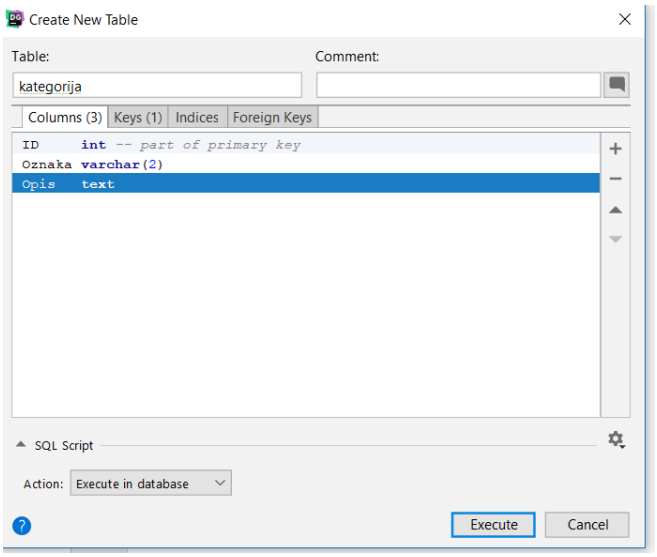


Slika 5: Unos potrebnih parametara

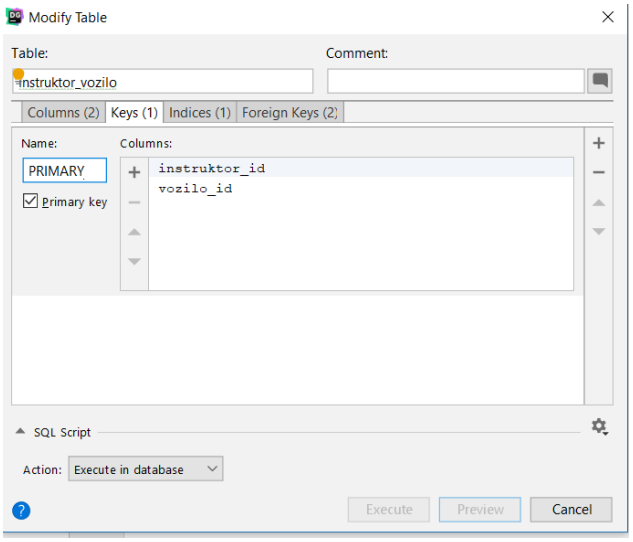
Za kreiranje nove tablice potreban je desni klik mišem na shemu autoskola te odabrati New iz ponuđenoga izbornika, a zatim kliknuti Table. Nakon toga otvara nam se nov skočni prozor u kome definiramo strukturu tablice, naziv, atribute, ključeve i tipove podataka.



Slika 6: Kreiranje nove tablice



Slika 7: Definiranje atributa tablice

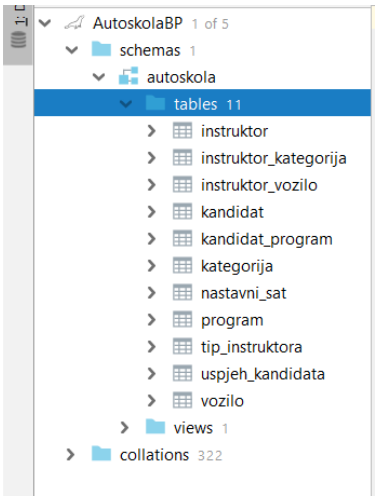


Slika 8: Kreiranje višekomponentnog primarnog ključa

Za kreiranje višekomponentnog primarnoga ključa u konzoli moguće je koristiti naredbu ADD CONSTRAINT ili samo CONSTRAINT prilikom kreiranja tablice, što je prikazano na primjeru kreiranja tablice instruktor_kategorija:

```
CREATE TABLE instruktor_kategorija (instruktor_id INT REFERENCES
instruktor(ID) ON DELETE CASCADE ON UPDATE CASCADE, kategorija_id INT
REFERENCES kategorija(ID) ON DELETE CASCADE ON UPDATE CASCADE,
CONSTRAINT instruktor_kategorija_pk PRIMARY KEY (instruktor_id,
kategorija_id));
```

Nakon što su kreirane sve tablice potrebne za bazu podatka moguće ih je sve vidjeti i u alatu i u konzoli što je vidljivo na slikama 9 i 10.



Slika 9: Završna struktura baze podataka u alatu DataGrip

```
MariaDB [autoskola]> SHOW TABLES;
+-----------------------+
| Tables_in_autoskola   |
+-----------------------+
| instruktor            |
| instruktor_kategorija |
| instruktor_vozilo     |
| kandidat              |
| kandidat_program      |
| kategorija            |
| nastavni_sat          |
| program               |
| tip_instruktora       |
| uspjeh_kandidata      |
| vozilo                |
+-----------------------+
11 rows in set (0.001 sec)
```

Slika 10: Završna struktura baze podataka vidljiva u konzoli

6.5. Implementacija ograničenja

Za svaku bazu podatka koja će valjano izvršavati svoju funkciju, od velike su važnosti razna ograničenja. U ovome dijelu rada pokazat ću kako dodati ograničenje te kako kreirati okidač.

6.5.1. CHECK CONSTRAINT

U odabranoj domeni autoškole najprije ću implementirati ograničenje koje će osigurati da vrijednost stupca Uspjeh tablice uspjeh_kandidata uvijek bude 'polozio' ili 'nije polozio'. To ću postići kreiranjem novoga ograničenja nad postojećom tablicom kombinacijom naredbi ALTER TABLE i ADD CONSTRAINT.

```
ALTER TABLE uspjeh_kandidata ADD CONSTRAINT CK_uspjeh_kandidata CHECK
(Uspjeh='nije polozio' OR uspjeh='polozio');
```

Za provjeru je li ograničenje doista dodano u odgovarajuće obliku moguće je koristiti SHOW CREATE TABLE, a rezultat takvoga upita vidljiv je na Slici 11.

```
SHOW CREATE TABLE uspjeh_kandidata;
```

```
| uspjeh_kandidata | CREATE TABLE `uspjeh_kandidata` (
  `kandidat_id` int(11) NOT NULL,
  `program_id` int(11) NOT NULL,
  `kategorija_id` int(11) NOT NULL,
  `Datum` date NOT NULL,
  `Uspjeh` varchar(20) NOT NULL,
  `Postotak_uspjesnosti` int(11) NOT NULL,
  PRIMARY KEY (`kandidat_id`,`program_id`,`kategorija_id`,`Datum`),
  KEY `uspjeh_kandidata_kategorija_ID_fk` (`kategorija_id`),
  KEY `uspjeh_kandidata_program_ID_fk` (`program_id`),
  CONSTRAINT `uspjeh_kandidata_kandidat_ID_fk` FOREIGN KEY (`kandidat_id`) REFERENCES `kandidat` (`ID`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `uspjeh_kandidata_kategorija_ID_fk` FOREIGN KEY (`kategorija_id`) REFERENCES `kategorija` (`ID`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `uspjeh_kandidata_program_ID_fk` FOREIGN KEY (`program_id`) REFERENCES `program` (`ID`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `CK_uspjeh_kandidata` CHECK (`Uspjeh` = 'nije polozio' or `Uspjeh` = 'polozio')
) ENGINE=InnoDB DEFAULT CHARSET=latin1 |
```

Slika 11: Rezultat upita SHOW CREATE TABLE

Na sličan način moguće je dodati još neka ograničenja po potrebi.

6.5.2. Okidači

Osim navedenoga ograničenja, nadalje ću pokazati kreiranje okidača koji su vrlo važni kod rada s bazom podataka. U nastavu ću kreirati 6 okidača koji će pokriti neke krajnje slučajeve pri radu s bazom podataka, a koje ne smijemo dozvoliti korisnicima.

Okidač u SUBP MariaDB kreira se korištenjem ključne riječi CREATE TRIGGER. Moguće ga je kreirati za razne događaje poput BEFORE INSERT, AFTER INSERT, BEFORE UPDATE, AFTER UPDATE, BEFORE DELETE, AFTER DELETE i slično. Nakon definiranja događaja koji će uzrokovati pokretanje okidača, potrebno je navesti tablicu nad kojom će se okidač kreirati i zatim slijedi tijelo okidača u kojemu se kodom specificira što će se dogoditi kada se okine okidač. Sve kreirane okidače moguće je pogledati izvršavanjem naredbe SHOW TRIGGERS.

Opis funkcionalnosti i slučaja okidanja okidača te sam kod okidača koje sam implementirala nad bazom podataka za potrebe autoškole naveden je u sljedećoj tablici:

Tablica 13: Okidači

OPIS OKIDAČA	KOD OKIDAČA

Ovaj okidač osigurava da kandidatu ne može nastavni sat održati instruktor koji nema dozvolu za obučavanje upisane kategorije. Varijable koje će se koristiti moraju se najprije deklarirati ključnom riječi DECLARE. Varijable tipa TABLE ne postoje u SUBP MariaDB stoga je za pohranu podataka potrebno kreirati privremenu tablicu korištenjem ključne riječi CREATE TEMPORARY TABLE, a istu je potrebno i obrisati u kodu kako bi se sa sljedećim pozivom mogla ponovno kreirati.	`CREATE TRIGGER provjera_instruktora_i_kategorije`
	`  BEFORE INSERT`
	`  ON nastavni_sat`
	`  FOR EACH ROW`
	`BEGIN`
	`  DECLARE kandidat_kategorija int;`
	`  CREATE TEMPORARY TABLE instruktorka(kat int);`
	`  select kandidat_program.kategorija_id into kandidat_kategorija from kandidat_program where kandidat_program.kandidat_id=new.kandidat_id;`
	`  insert into instruktorka(kat) select instruktor_kategorija.kategorija_id from instruktor_kategorija where instruktor_kategorija.instruktor_id=new.instruktor_id;`
	`  if not  exists(select kat from instruktorka where kat=kandidat_kategorija) then`
	`  drop temporary table instruktorka;`
	`  SIGNAL SQLSTATE '70002' SET MESSAGE_TEXT = 'Ne možete dodijeliti kandidatu instruktora koji ne podučava kategoriju koju je upisao kandidat';`
	`end if ;`
	`drop temporary table instruktorka;`
	`END;`

Ovaj okidač osigurava da kandidat ne može polagati ispit iz upravljanja prije nego je položio ispit iz propisa.	`create TRIGGER provjera_propisi_upravljanje`
	`  BEFORE INSERT`
	`  ON kandidat_program`
	`  FOR EACH ROW`
	`BEGIN`
	`  declare program int;`
	`  declare program_new int;`

	set program=1;
	select kandidat_program.program_id into program_new from kandidat_program where
	kandidat_program.kandidat_id=new.kandidat_id and kandidat_program.program_id=new.program_id and kandidat_program.kategorija_id=new.kategorija_id;
	if program_new=2 and not exists(select * from uspjeh_kandidata where uspjeh_kandidata.kandidat_id=new.kandidat_id and uspjeh_kandidata.Uspjeh='polozio' and uspjeh_kandidata.program_id=program and uspjeh_kandidata.kategorija_id=new.kategorija_id) then
	SIGNAL SQLSTATE '70002' SET MESSAGE_TEXT = 'Kandidat nije položio propise za navedenu kategoriju';
	end if ;
	
	END;

Ovaj okidač osigurava da kandidat ne može polagati ispit prije nego odradi potrebne sate obuke predviđene programom	create TRIGGER provjera_odrađenih_sati
	BEFORE INSERT
	ON uspjeh_kandidata
	FOR EACH ROW
	BEGIN
	declare potrebno_sati int;
	declare trenutno_sati int;
	
	select program.Broj_sati into potrebno_sati from program where program.ID=new.program_id;
	select sum(nastavni_sat.Trajanje) into trenutno_sati from nastavni_sat where nastavni_sat.kandidat_id=new.kandidat_id and nastavni_sat.program_id=NEW.program_id and nastavni_sat.kategorija_id=new.kategorija_id;
	
	if potrebno_sati>trenutno_sati then
	SIGNAL SQLSTATE '70002' SET MESSAGE_TEXT =
	'Kandidat nije odradio potrebne sate obuke';
	end if ;
	
	END;

Ovaj okidač osigurava da instruktor predavač ne može održati sat predavanja i obrnuto.	create TRIGGER provjera_instruktora_propisa BEFORE INSERT ON nastavni_sat FOR EACH ROW BEGIN declare instruktor_tip int;

```
  declare program_new int;


  select instruktor.tip_id into instruktor_tip from
instruktor where instruktor.ID=new.instruktor_id;
  select  nastavni_sat.program_id into program_new
from nastavni_sat where
nastavni_sat.program_id=NEW.program_id;


  if program_new=1 and instruktor_tip!=1 then
  SIGNAL SQLSTATE '70002' SET MESSAGE_TEXT =
  'Instruktor vožnje ne može održati predavanje';
  elseif program_new=2 and instruktor_tip=1 then
 SIGNAL SQLSTATE '70002' SET MESSAGE_TEXT =
  'Predavač ne može održati sat vožnje';
end if ;


END;
```

Ovaj okidač osigurava da se datum polaganja ispita, ukoliko je kandidat položio, upiše kao datum završetka upisanog programa

```
create TRIGGER upis_datuma_završetka
  after INSERT
  ON uspjeh_kandidata
  FOR EACH ROW
BEGIN
  declare uspjeh_kandidat varchar(10);
  declare datum_new date;


  select uspjeh_kandidata.Uspjeh into uspjeh_kandidat
from uspjeh_kandidata where
uspjeh_kandidata.kandidat_id=new.kandidat_id and
uspjeh_kandidata.program_id=NEW.program_id and
uspjeh_kandidata.kategorija_id=NEW.kategorija_id;
    select  uspjeh_kandidata.Datum into datum_new from
uspjeh_kandidata where
uspjeh_kandidata.kandidat_id=NEW.kandidat_id and
uspjeh_kandidata.program_id=NEW.program_id and
uspjeh_kandidata.kategorija_id=NEW.kategorija_id;


  if uspjeh_kandidat='polozio' then
    update kandidat_program set
datum_završetka=datum_new where
kandidat_program.kandidat_id=NEW.kandidat_id and
kandidat_program.program_id=NEW.program_id;


end if ;


END;
```

Ovaj okidač osigurava da kandidat ne može polagati ispit iz kategorije koju nije upisao. U primjeru je vidljiv način uporabe aliasa za naziv tablice kako se ne bi puni naziv stalno pisao.

```
create TRIGGER provjera_uspjeh_kategorija
  before INSERT
  ON uspjeh_kandidata
  FOR EACH ROW
BEGIN
  declare kandidat_kategorija int;
  create  temporary table kandidatkat (kat int);


  insert  into  kandidatkat(kat) select
kandidat_program.kategorija_id from kandidat_program
kp where kp.program_id=new.program_id and
kp.kandidat_id= NEW.kandidat_id;
  select kp.program_id into kandidat_kategorija from
kandidat_program kp where
kp.kandidat_id=NEW.kandidat_id and
kp.kategorija_id=new.kategorija_id and
kp.program_id=new.program_id;


  if not exists(select  kat from kandidatkat where
kat=kandidat_kategorija) then
   SIGNAL SQLSTATE '70002' SET MESSAGE_TEXT =
   'Kandidat ne može izaći na ispit iz kategorije koju
nije upisao programom';


end if ;


END;
```

6.6. Upiti nad kreiranom bazom podataka

Nad kreiranom bazom podataka, koju sam prethodno popunila podacima, moguće je kreirati mnoštvo različitih upita. Kao i u svakom SUBP-u, tako i u SUBP MariaDB razlikuju se jednostavni upiti kojima dohvaćamo podatke jedne tablice, te složeni upiti kojima kombiniramo podatke više tablica. U nastavku ću pokazati nekoliko primjera.

6.6.1. Jednostavni upiti

Recimo da želimo ispisati imena i prezimena svih instruktora autoškole, to možemo postići izvršavanjem sljedećeg upita:

```
MariaDB [autoskola]> SELECT ime, prezime FROM instruktor;
+-------+---------+
| ime   | prezime |
+-------+---------+
| Pero  | Peric   |
| Anto  | Antic   |
| Marko | Maric   |
+-------+---------+
3 rows in set (0.001 sec)
```

Slika 12: Rezultat upita nad tablicom instruktor

Kao što je vidljivo potrebno je samo navesti imena stupaca čije podatke želimo prikazati te u klauzuli FROM navesti naziv tablice koja će biti izvor podataka. Na jednak način možemo ispisati i registarske oznake svih vozila autoškole, te još mnogo toga.

```
MariaDB [autoskola]> SELECT registarske_oznake FROM vozilo;
+--------------------+
| registarske_oznake |
+--------------------+
| ZP2587             |
| UK258974           |
+--------------------+
2 rows in set (0.007 sec)
```

Slika 13: Rezultat upita nad tablicom vozilo

6.6.2. Složeni upiti

Za rad s bazom podataka od veće su važnosti složeni upiti kojima povezujemo podatke iz više tablica ne temelju ključeva. Složenim upitima moguće je dobiti razne statističke informacije koje se tiču podataka pohranjenih u bazi. U složenim upitima moguće je podatke sortirati i grupirati, upotrebljavati razne agregirajuće funkcije i još mnogo toga.

Za početak recimo da želimo ispisati sve instruktore autoškole, te nazive tipova kojima oni pripadaju. To možemo postići na sljedeći način:

```
MariaDB [autoskola]> SELECT i.*, t.naziv FROM instruktor i JOIN tip_instruktora t ON i.tip_id=t.ID;
+----+-------+---------+-------------+-------------------+---------------+--------+--------------------+
| ID | Ime   | Prezime | OIB         | Email             | Broj_telefona | tip_id | naziv              |
+----+-------+---------+-------------+-------------------+---------------+--------+--------------------+
|  1 | Pero  | Peric   | 03265478945 | pperic@gmail.com  | 0635897412    |      1 | Predavac           |
|  2 | Anto  | Antic   | 32145698745 | NULL              | 123456        |      1 | Predavac           |
|  3 | Marko | Maric   | 36521459875 | mmaric@gmail.com  | 0996874589    |      2 | Instruktor vožnje  |
+----+-------+---------+-------------+-------------------+---------------+--------+--------------------+
3 rows in set (0.007 sec)
```

Slika 14: Rezultat složenoga upita nad tablicama instruktor i tip_instruktora

U primjeru je vidljivo spajanje tablica korištenjem ključne riječi JOIN.

Nadalje, recimo da želimo ispisati ukupan broj pohađanih nastavnih sati za određenoga kandidata (pretpostavimo za kandidata najprije s ID-jem 1, a zatim i po njegovom prezimenu).

```
MariaDB [autoskola]> SELECT SUM(ns.trajanje) as broj_sati FROM nastavni_sat ns, kandidat k WHERE ns.kandidat_id=k.ID AND k.ID=1;
+-----------+
| broj_sati |
+-----------+
|         4 |
+-----------+
1 row in set (0.005 sec)

MariaDB [autoskola]> SELECT SUM(ns.trajanje) as broj_sati FROM nastavni_sat ns, kandidat k WHERE ns.kandidat_id=k.ID AND k.prezime='Anic';
+-----------+
| broj_sati |
+-----------+
|         4 |
+-----------+
1 row in set (0.001 sec)
```

Slika 15: Rezultat složenoga upita nad tablicama kandidat i nastavni_sat

Pretpostavimo da želimo vidjeti koliko nastavnih sati je zabilježeno za kojega kandidata. Najprije samo za kandidate koji su bili na barem jednom satu, a zatim i za kandidate koji nemaju zabilježen niti jedan sat. U upitu ću kombinirati naredbe COUNT i GROUP BY.

```
MariaDB [autoskola]> SELECT k.*, COUNT(ns.ID) AS broj_sati FROM kandidat k, nastavni_sat ns WHERE k.ID=ns.kandidat_id GROUP BY 1;
+----+-----+---------+-------------+---------------+-------+--------------+-----------+
| ID | Ime | Prezime | OIB         | Datum_ro-enja | Email | Kontakt_broj | broj_sati |
+----+-----+---------+-------------+---------------+-------+--------------+-----------+
|  1 | Ana | Anic    | 12345678998 | 1970-01-02    | NULL  | 0996366073   |         2 |
+----+-----+---------+-------------+---------------+-------+--------------+-----------+
1 row in set (0.001 sec)

MariaDB [autoskola]> SELECT k.*, COUNT(ns.ID) AS broj_sati FROM kandidat k LEFT JOIN nastavni_sat ns ON k.ID=ns.kandidat_id GROUP BY 1;
+----+-------+----------+-------------+---------------+---------------+--------------+-----------+
| ID | Ime   | Prezime  | OIB         | Datum_ro-enja | Email         | Kontakt_broj | broj_sati |
+----+-------+----------+-------------+---------------+---------------+--------------+-----------+
|  1 | Ana   | Anic     | 12345678998 | 1970-01-02    | NULL          | 0996366073   |         2 |
|  2 | Karlo | Karlovic | 98745625142 | 1974-01-18    | NULL          | 023654789    |         0 |
|  3 | Mirko | Mirkovic | 12365478956 | 1970-01-17    | mmirkovic.com | 123654789    |         0 |
+----+-------+----------+-------------+---------------+---------------+--------------+-----------+
3 rows in set (0.001 sec)
```

Slika 16: Uporaba naredbi COUNT i GROUP BY

Prethodne rezultate moguće je i sortirati recimo po prezimenu silazno.

```
MariaDB [autoskola]> SELECT k.*, COUNT(ns.ID) AS broj_sati FROM kandidat k LEFT JOIN nastavni_sat ns ON k.ID=ns.kandidat_id GROUP BY 1
ORDER BY k.prezime DESC;
+----+-------+----------+-------------+---------------+---------------+--------------+-----------+
| ID | Ime   | Prezime  | OIB         | Datum_ro-enja | Email         | Kontakt_broj | broj_sati |
+----+-------+----------+-------------+---------------+---------------+--------------+-----------+
|  3 | Mirko | Mirkovic | 12365478956 | 1970-01-17    | mmirkovic.com | 123654789    |         0 |
|  2 | Karlo | Karlovic | 98745625142 | 1974-01-18    | NULL          | 023654789    |         0 |
|  1 | Ana   | Anic     | 12345678998 | 1970-01-02    | NULL          | 0996366073   |         2 |
+----+-------+----------+-------------+---------------+---------------+--------------+-----------+
3 rows in set (0.001 sec)
```

Slika 17: Sortiranje rezultata upita

Na sličan način moguće je izgraditi još mnoštvo drugih upita nad bazom podataka u ovisnosti o tome što nam je potrebno.

6.7. Pogledi

Pogledi *(engl. Views)* nam mogu biti korisni ukoliko neki složeni upit želimo izvršiti bez da ga ponovno pišemo. Recimo da u našem primjeru želimo ispisati sve kandidate koji su uspješno položili ispit. Za takav upit možemo kreirati pogled, a onda pomoću pogleda ispisati podatke o kandidatima. To bi učinili na sljedeći način:

```
CREATE VIEW polozili AS select k.ime, k.prezime from kandidat k,
uspjeh_kandidata uk where k.id=uk.kandidat_id and
uk.uspjeh='polozio';

SELECT * FROM polozili;
```

6.8. Virtualni stupci

Kako u odabranoj bazi podataka nema potrebe za virtualnim stupcima, odnosno nemamo takve podatke koje bi trebalo računati, za potrebe prikazivanja primjera kreiranja virtualnog stupca kreirati ću novu tablicu račun u kojoj će se nalaziti podaci o artiklu, količini, cijeni i iznosu. Vrijednost stupca iznos računat će se kao količina*cijena, odnosno imat će ulogu virtualnog stupca.

```
create table racun (Artikl varchar(20), Kolicina int, Cijena double,
Iznos double as (Kolicina * Cijena) persistent);
```

Zatim ću umetnuti jedan redak u kreiranu tablicu:

```
insert into racun(Artikl, Kolicina, Cijena) values ('stol',3,50);
```

i na kraju ispisati podatke tablice, te uočiti kako se stupac iznos zaista izračunao kao količina*cijena:

```
MariaDB [autoskola]> SELECT * FROM racun;
+--------+----------+--------+-------+
| Artikl | Kolicina | Cijena | Iznos |
+--------+----------+--------+-------+
| stol   |        3 |     50 |   150 |
+--------+----------+--------+-------+
1 row in set (0.007 sec)
```

Slika 18: Prikaz rezultata upita nad tablicom s kreiranim virtualnim stupcem

```
MariaDB [autoskola]> HANDLER racun OPEN;
Query OK, 0 rows affected (0.000 sec)

MariaDB [autoskola]> HANDLER racun READ FIRST;
+--------+----------+--------+-------+
| Artikl | Kolicina | Cijena | Iznos |
+--------+----------+--------+-------+
| stol   |        3 |     50 |   150 |
+--------+----------+--------+-------+
1 row in set (0.001 sec)
```

Slika 19: Ispis sadržaja tablice korištenjem NoSQL naredbe HANDLER

6.9. Napredno pretraživanje teksta

Za napredno pretraživanje teksta najprije je potrebno kreirati FULLTEXT INDEX nad stupcem koji je tipa TEXT. Primjer kreiranja ovakvoga indeksa i na kraju napredno pretraživanje pokazati ću nad tablicom kategorija gdje je stupac Opis tipa TEXT. Dakle, za kreiranje indeksa koristi se sljedeća naredba:

```
create fulltext index i1 on kategorija(Opis);
```

i zatim je moguće pretraživanje:

```
MariaDB [autoskola]> SELECT * FROM kategorija WHERE MATCH (opis) AGAINST ('motorna');
+----+--------+-----------------------------------------+
| ID | Oznaka | Opis                                    |
+----+--------+-----------------------------------------+
|  2 | B      | motorna vozila                          |
|  3 | C1     | motorna vozila osim onih iz kategorije D1 |
|  4 | C      | motorna vozila                          |
+----+--------+-----------------------------------------+
```

Slika 20: Rezultat naprednog pretraživanja teksta

7. Zaključak

Sustav za upravljanje bazama podataka MariaDB, dakle, nije uzalud najbrže rastući nerelacijski SUBP na tržištu. Svoj rast i razvoj duguje stalnom naporu svojih autora na unapređivanju postojećih karakteristika te na stvaranju novih. U mnogim situacijama upravo ovaj SUBP pokazao se odličnim izborom što potvrđuju i brojne velike i poznate tvrtke koje ga upotrebljavaju. Najjača snaga ovoga sustava su njegove performanse, sigurnost, otvorenost prema svim korisnicima, dostupnost te pouzdanost. Lakoća rada na bazi podataka dodatna je prednost ovoga sustava, stoga nije teško prijeći s nekoga drugoga sustava na ovaj, pogotovo ukoliko se radi o sustavu MySQL. Upravo sa MySQL sustavom je nejčešće uspoređivan zbog njihove kompatibilnosti, no ipak postoje razlike između njih. Osnovne karakteristike kojima SUBP MariaDB proširuje funkcionalnosti MySQL-a jesu te što omogućava podršku za statističku obradu podataka, omogućava rad s geografskim podacima te sadrži različite funkcije za obradu teksta. Još jedna od razlika su dinamički stupci koje sustav MySQL ne podržava, a oni su jedna od osnovnih karakteristika SUBP MariaDB. Osim toga, razlika je u performansama gdje pobjeđuje ponovno SUBP MariaDB. Ukoliko koristimo varijantu SQL-a kao jezika oba sustava i MariaDB i MySQL mogu izvršavati isti izvršni kod. Iz rada se da zaključiti da su brojne prednosti SUBP-a MariaDB u odnosu na njegove konkurente te da će one u budućnosti zasigurno još više rasti.

Inovativne ideje njegovih autora konstantno doprinose i doprinositi će razvoju ovoga sustava za upravljanje bazama podataka, a ne će čuditi ukoliko uskoro bude i jedan od najkorištenijih sustava za upravljanje bazama podataka na svijetu.

Popis literature

- Carić T., Buntić M., *„Uvod u relacijske baze podataka“*, Fakultet prometnih znanosti, Sveučilište u Zagrebu, 2015
- Cvetemirov Ivanov I., *„Reasons to migrate to MariaDB“*, ResearchGate, 2019, dostupno 20.5.2019., preuzeto s: https://www.researchgate.net/publication/331074421_Reasons_to_migrate_to_MariaDB
- Goel A., *„MariaDB vs MySQL“*, Hackr.io, 2019, dostupno 20.5.2019., preuzeto s: https://hackr.io/blog/mariadb-vs-mysql
- Maleković M., Rabuzin K., *„Uvod u baze podataka“*, Fakultet organizacije i informatike, Varaždin, 2016
- MariaDB, *„NoSQL“*, 2019, dostupno 17.6.2019., preuzeto s: https://mariadb.com/kb/en/library/nosql/
- MariaDB, „*SQL Statements*“, dostupno 17.6.2019., preuzeto s: https://mariadb.com/kb/en/library/sql-statements/
- MariaDB, „*Data Types*“, 2019., dostupno 17.6.2019., preuzeto s: https://mariadb.com/kb/en/library/data-types/
- MySQLTUTORIAL, *„MariaDB vs. MySQL“*, 2019, dostupno 20.5.2019, preuzeto s: http://www.mysqltutorial.org/mariadb-vs-mysql/
- Rabuzin K., „*SQL: napredne teme*“, Fakultet organizacije i informatike, Varaždin, 2014
- Rabuzin K., „*Uvod u SQL*“, Fakultet organizacije i informatike, Varaždin, 2011
- Sarig M., *„MariaDB Vs MySQL In 2019: Compatibility, Performance, And Syntax“*, Panoply, 2019, dostupno 20.5.2019., preuzeto s: https://blog.panoply.io/a-comparative-vmariadb-vs-mysql
- Shay T., *„MariaDB vs MySQL – Comparing MySQL 8.0 with MariaDB 10.3“*, EverSQL, 2018, dostupno 20.5.2019., preuzeto s: https://www.eversql.com/mariadb-vs-mysql/
- Solid IT, *„System Properties Comparison MariaDB vs. MySQL“*, DB-Engines, 2019, dostupno 20.5.2019., preuzeto s: https://db-engines.com/en/system/MariaDB%3BMySQL

- Stefanović K., *„Razmatranje nerelacionih karakteristika SUBP MariaDB na primeru razvoja web aplikacije za razmenu poruka i dokumenata“*, Master rad, Matematički fakultet, Univerzitet u Beogradu, Beograd, 2018, dostupno 20.5.2019., preuzeto s: http://www.racunarstvo.matf.bg.ac.rs/MasterRadovi/2014_Kristina_Stefanovic/rad.pdf

- Tivanovac M., *„Modeliranje nerelacijskih baza podataka“*, Završni rad, Fakultet elektrotehnike, računarstva i informacijskih tehnologija, Sveučilište Josipa Jurja Strossmayera u Osijeku, Osijek, 2016, dostupno 20.5.2019., preuzeto s: https://repozitorij.etfos.hr/islandora/object/etfos:970/preview

- TrustRadius, „*MariaDB Reviews*“, 2018, dostupno 17.6.2019., preuzeto s: https://www.trustradius.com/products/mariadb/reviews

- Wikipedia, *„Baza podataka“*, 2019, dostupno 20.5.2019., preuzeto s: https://hr.wikipedia.org/wiki/Baza_podataka

- Wikipedia, *„MariaDB“*, 2019, dostupno 20.5.2019., preuzeto s: https://en.wikipedia.org/wiki/MariaDB

Popis slika

Popis tablica