SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Matija Vuk

Sustav za upravljanje bazom podataka MySQL

ZAVRŠNI RAD

Varaždin, 2019.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Matija Vuk

Matični broj: 44048/15–R

Studij: Informacijski sustavi

Sustav za upravljanje bazom podataka MySQL

ZAVRŠNI RAD

Mentor/Mentorica:

Prof. dr. sc. Maleković Mirko

Varaždin, rujan 2019.

Matija Vuk

Izjava o izvornosti

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

Autor potvrdio prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

Tema ovog završnog rada je sustav za upravljanje bazom podataka MySQL (u daljnjem kontekstu SUBP) no prije same obrade ovo sustava, potrebno je objasniti što su uopće baze podataka te koje su razlike u vrstama baze podataka. Kako bi obradili SUBP MySQL, spomenut ćemo povijest istog te verzije, a tada ćemo obraditi arhitekturu, kao što su naredbe te tipovi podataka. Rad također obuhvaća izradu aplikacije koja koristi MySQL bazu podataka za skladištenje informacija. Aplikacija je rađena u Symfony frameworku te za interakciju za bazom se koristi Doctrine ORM.

Ključne riječi: SUBP, sustav, baza podataka, MySQL, Symfony, Doctrine, PHP

Sadržaj

1. Uvod

Baze podataka nezaobilazan su dio današnje svakodnevice. Koriste se u svim okolinama, a ponajviše u poslovnim, kako bi se lakše organizirali podaci za daljnju obradu ili prikaz. Iako često puta ne razmišljamo o tome, naše svakodnevne aktivnosti uključuju interakciju s bazama podataka. Uzmimo samo primjer korištenja društvenih mreža, koje koristimo na dnevnoj bazi, kod svakog pokretanja aplikacije koristi se baza podataka kako bi se utvrdio identitet korisnika koji pokušava pristupiti određenom sadržaju.

Kako bi mogli upravljati bazom podataka, potreban nam je alat za to. Takav alat zove se „Sustav za upravljanje bazom podataka“ (SUBP) tj. „Database management system“ (DBMS), a jedan od najzastupljenijih je MySQL. Ovaj alat otvorenog koda pokreće se na poslužitelju, a model baze podataka koji koristi je relacijski. Zbog svoje relativno jednostavne sintakse, velike podrške zajednice te dobrih performansa, MySQL odabir je mnogih svjetskih kompanija od kojih možemo izdvojiti GitHub, YouTube, NASA, Airbus, CERN, itd.[1]

U ovome radu obradit ćemo MySQL kroz aplikacijsku domenu koja će koristiti bazu podataka, te objasniti rad i interakciju aplikacije s bazom podataka kroz nekoliko cjelina. Za aplikacijsku domenu koristiti ću Symfony framework koji će pogoniti PHP (verzija 7.3) s integriranim serverom, a za interakciju s bazom Doctrine ORM. Aplikacija je napravljena kao sustav koji pomaže studentima tokom studiranja u smislu lakšeg pronalaženja informacija o upisanim kolegijima. Nakon uspješne registracije, korisnik mora unijeti upisane kolegije, te nakon toga može objaviti novosti o pojedinom upisanom kolegiju, prenijeti Word datoteku ili objaviti da traži ili nudi instrukcije iz odabranog kolegija. Također, svaki korisnik može glasati koliko je informacija korisna tako da klikne na „korisno“ ili „nije korisno“ na samoj objavi..

2. Metode i tehnike rada

Prvi dio ovog završnog rada posvećen je obradi bazama podataka općenito, a za cilj ima upoznati korisnika s osnovama baza podatak kako bi drugi i treći dio bio jasniji. U obradi baza podataka objasniti ću arhitekturu relacijskih baza podataka jer ću se i sam koristiti istim, nakon toga bitno je objasniti što je zapravo sustav za upravljanje bazama podataka te čemu on služi.

Drugi dio rada odnosi se na samu temu, sustav za upravljanje bazama podataka MySQL. Da bi se upoznali sa MySQL-om obradit ću kratku povijest istog, nakon toga arhitekturu te sintaksu i tipove podataka koje ono podržava.

Zadnji dio ovog završnog rada odnosi se na primjenu MySQL-a za razvoj odabrane aplikacije. Aplikacija je namijenjena da pomogne studentima jednostavnije pronaći tražene informacije za odabrane kolegije. Za razvoj ove aplikacije istaknuo bih da sam između ostalog koristio PHP programski jezik, Symfony framework, Doctrine ORM, PHPStorm kao integrirano razvojno okruženje te MySQL Workbench za izradu ER modela baze podataka.

3. Baze podataka

U svakodnevnom životu vrlo često imamo potrebu organizirati određene informacije kako bi im kasnije lakše pristupili, odnosno koristili. Kao primjer takve organizacije možemo navesti telefonski imenik, kartoteku te ostale razne evidencije. No, ovakve organizacije podataka su vrlo ograničavajuće te nam ne pružaju dovoljno mogućnosti za manipulacijom podataka, u tu svrhu koristimo baze podataka.

U knjizi „Uvod u relacijske baze podataka“ (2015) autori navode da je bazu podataka potrebno promatrati kao zbirku zapisa podataka pohranjenih na računalu te da mora biti lako dostupna korisnicima i aplikacijama koje imaju dozvoljen pristup toj bazi podataka. Nadalje, sve podatke unutar baze podataka potrebno je pohraniti bez nepotrebne zalihosti.

Bazu podataka možemo opisati kao organiziranu i uređenu cjelinu međusobno povezanih podataka bez nepotrebne zalihosti. No, tim podacima je potrebno manipulirati, za tu svrhu se koristi sustav za upravljanje bazom podataka (eng. *DBMS*). Kako navodi Robert Manger (2010) u knjizi „Osnove projektiranja baze podataka“, sustav za upravljanje bazom podataka fizički oblikuje prikaz baze te je poslužitelj (eng. *server*) same baze podataka. SUBP obavlja sve operacije s podacima u ime klijenta, ukoliko klijent ima potrebne ovlasti za pristup tim podacima.

Kako bi implementirali samu bazu podataka, potrebno je odrediti skup pravila kako će izgledati logička struktura podataka, tu dolazimo do pojma modela podataka. Postoji više modela podataka, no najzastupljeniji je relacijski model podataka.

3.1. Relacijski podatkovni model baze podataka

Ovaj podatkovni model baze podataka potječe iz Engleske, točnije, razvio ga je engleski matematičar Edgar Frank Codd, a prezentiran je davne 1970. godine u članku istog autora *“A Relational Model of Data for Large Shared Data Banks”*. Zanimljivo je što, iako je u to vrijeme Edgar radio u IBM-u, zbog konzervativnosti IBM-a i ustrajnosti korištenja već provjerenog, iako zastarjelog hijerarhijskog modela, prva tvrtka koja je implementirala relacijski model podataka je bila Oracle.[2]

Relacijski model baze podataka pohranjuje podatke u tablice, odnosno svaka tablica sadrži podatke o jednom entitetu. Svaka takva tablica se sastoji od redaka i stupaca. Unutar redaka pohranjujemo podatke o jednom entitetu. Entitet može biti bilo kakva informacija koju možemo opisati. Unutar stupaca pohranjujemo atribute, odnosno opis entiteta. Naravno, jedan

entitet može imati više atributa, no mora sadržavati primarni ključ koji mora biti jedinstven za svaki entitet, odnosno redak.

ID	Naziv uloge
1	ADMIN
2	MODERATOR
3	KORISNIK

Tablica 1 Primjer tablice

Primarni ključ može služiti za povezivanje s drugim tablicama, te se on unutar povezane tablice naziva vanjski ključ. Kod ovakvih tablica, vanjski ključ ne smije biti nepoznata vrijednost (eng. *NULL*). Nadalje, veze između dva povezana entiteta, odnosno između dvije tablice mogu biti:

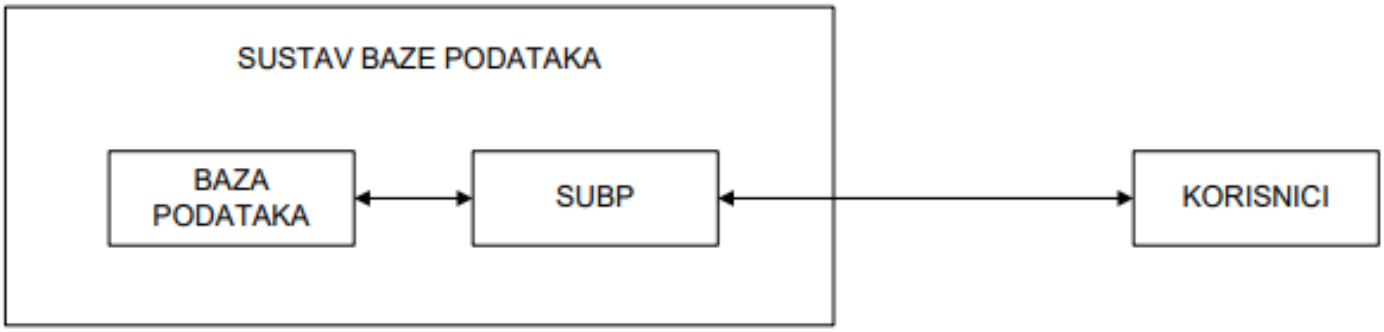
ID	Naziv uloge	Ime	Prezime	Uloga_ID
1	mvuk@foi.hr	Matija	Vuk	1
2	pperic@foi.hr	Pero	Perić	2
3	iivic@foi.hr	Ivo	Ivić	3

Tablica 2 Primjer tablice s vanjskim ključem

1. 1:1; zapis jedne tablice je vezan za samo jedan zapis iz druge tablice. Ovo se obično ne koristi u praksi, već se upotrebljava kada želimo dobiti više manjih tablica.
2. 1:M; veza jedan prema više se koristi najčešće. Jedan zapis iz prve tablice može biti povezan sa više entiteta iz druge tablice, dok jedan entitet iz druge tablice može imati samo jedan entitet iz prve tablice.
3. M:M; veza više prema više povezuje dvije tablice tako da sprema njihove primarne ključeve u treću tablicu. Npr: jedan student upisuje više kolegija na fakultetu, te je na jedan kolegij upisano više studenata. Kod ovakve situacije je potrebno uvesti novu, treću tablicu u kojoj će biti spremljeni primarni ključevi iz tablice studenti te primarni ključevi iz tablice kolegiji.[3]

3.2. Sustav za upravljanje bazama podataka

Sustav za upravljanje bazama podataka (*eng. Database Management System*) omogućava nam lakše manipuliranje podacima u bazi. Ovaj sustav možemo opisati kao posrednika između klijenta (ili aplikacije) i baze podataka. Ukoliko klijent ima validne pristupne podatke, pomoću SUBP-a može pristupiti određenim podacima unutar baze.



Slika 1 Struktura sustava baze podataka [8]

Sustav za upravljanje bazama podataka pomoću različitih naredbi može manipulirati svim podacima unutar baze kao što su upis u bazu podataka, čitanje podataka, izmjena podataka te kreiranje novih. No to su tek osnovne funkcionalnosti sustava, on nam pomoću tih funkcionalnosti omogućuje lakše i kvalitetnije manipuliranje podacima u smislu kompliciranijih uvjeta, kao što je izvršavanje neke radnje nad podacima nakon određenog događaja. Ovakve funkcije nazivamo okidači (*eng. trigger)*.

Postoje mnogi sustavi za upravljanje bazom podataka, no neki od najpoznatijih koje bi htio istaknuti su:

- MySQL
- PostgreSQL
- Oracle
- Microsoft Access
- SQL server

3.3. SQL

Strukturirani upitni jezik (*eng. Structured Query Language*) pojavio se kao odgovor za spremanje podataka u relacijskim bazama podataka. IBM je 1974. godine objavio članak pod nazivom SEQUEL (*eng. Struciured English Query Language*). Cilj objave tog članka bio je približavanje relacijskih baza podataka korisnicima, pa je tako i sam SQL prirodan i jednostavan za korištenje. Kako je upravo ta kratica već bila zaštitni znak zrakoplovne kompanije Hawker Siddeley, ista je zahtijevala da se to ime ne koristi, pa je kratica strukturiranog upitnog jezika promijenjena u današnji SQL. [2]

SQL naredbe mogu se klasificirati u nekoliko osnovnih skupina (Rabuzin, 2011).

- DML (*eng. Data Manipulation Language*); u ovu skupinu spadaju osnovne naredbe za upravljanje podacima, odnosno: INSERT, DELETE, UPDATE. Dakle, ovim naredbama možemo manipulirati podacima tako da ih upisujemo u određeni entitet unutar baze, brišemo ili ažuriramo
- DDL (*eng. Data Definition Language*); ova skupina naredbi je malo kompliciranija od prethodnih. Naime, u ovu skupinu spadaju naredbe pomoću kojih kreiramo tablice, poglede ili indekse, odnosno, naredbe koje spadaju u ou skupinu su: CREATE, DROP, ALTER
- DQL (*eng. Data Query Language*); ova skupina je vrlo jednostavna što se tiče naredbi jer u nju spada SELECT naredba, pomoću koje dohvaćamo podatke iz baze te kojom započinje velika većina upita nad podacima

4. Sustav za upravljanje bazom podataka MySQL

Sustav za upravljanje bazom podataka MySQL jedan je od najpopularnijih sustava za upravljanje bazama podataka. Jedan od razloga za to je činjenica što je pogodan i za male, jednostavne baze podataka te i za kompleksne baze podataka jer može sadržavati i preko 50 milijuna redaka u jednoj tablici. Nadalje, jednostavan je za korištenje zbog toga što koristi standardni oblik već poznatog i prije spomenutog SQL-a. Kako je originalno dizajniran da manipulira kompleksnim i velikim količinama podacima, MySQL je vrlo brz sustav te je bitno napomenuti da je otvorenog koda, što znači da naprednim korisnicima omogućava slobodu korištenja, u smislu izrade kopije i modificiranje programa. [4]

4.1. Povijest

MySQL je razvila švedska tvrtka MySQL AB, koja je osnovana 1995. godine. Sa 400 zaposlenih u 25 zemalja jedna je od najvećih open source tvrtki na svijetu, no zanimljivo je to što je čak 70% zaposlenika radilo od kuće.

MySQL AB je 2008. godine kupila tvrtka Sun, no ubrzo su započele poteškoće te je već 2 godine kasnije, 2010. godine, otkupila tvrtka Oracle. [5]

Možemo reći kako je MySQL prethodnik mSQL sustava. Naime, mSQL sustav je bio zamišljen da da manipulira tablicama baze podataka, no zbog njegove nedovoljne brzine izvršavanja tražila su se daljnja rješenja. To je rezultiralo novim SQL sučeljem, no API sučelje je ostalo isto kao i kod mSQL-a. [4]

MySQL je nazvan po kćeri jednoga od osnivača (Monty Widenius), My.

Slika 2 MySQL logo [4]

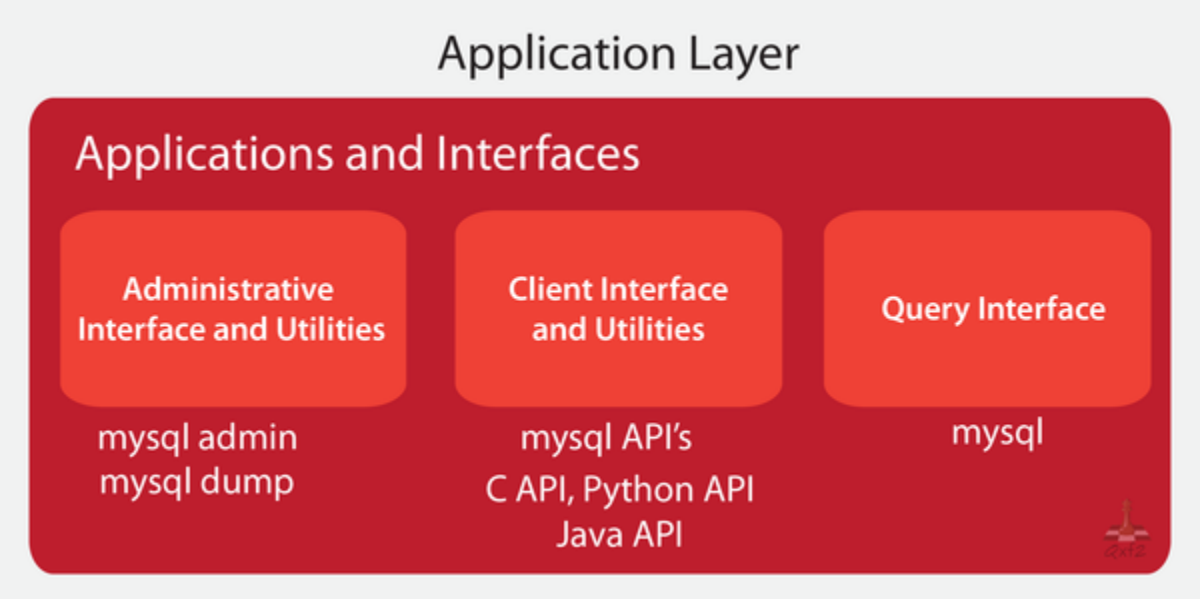
4.2. Arhitektura

MySQL je kompleksan sustav sa brojnim funkcionalnostima. Zbog svoje fleksibilnosti može raditi u raznim okruženjima te na raznim hardverima pošto ne zahtjeva puno resursa za sam rad. Može se koristiti za jednostavne aplikacije sa nekoliko tablica, no zbog svoje brzine, također se može koristiti i za velike baze podataka. Arhitektura MySQL-a je višeslojna, sastoji se od tri sloja;

1. Aplikacijski sloj
2. Logički sloj
3. Fizički sloj

4.2.1. Aplikacijski sloj

Ovaj sloj predstavlja interakciju korisnika sa MySQL sustavom. Svi servisi za spajanje, autentikaciju i sigurnost nalaze se u ovome sloju. U sljedećoj su slici prikazane tri glavne komponente; administrator, klijent i sučelje upita.

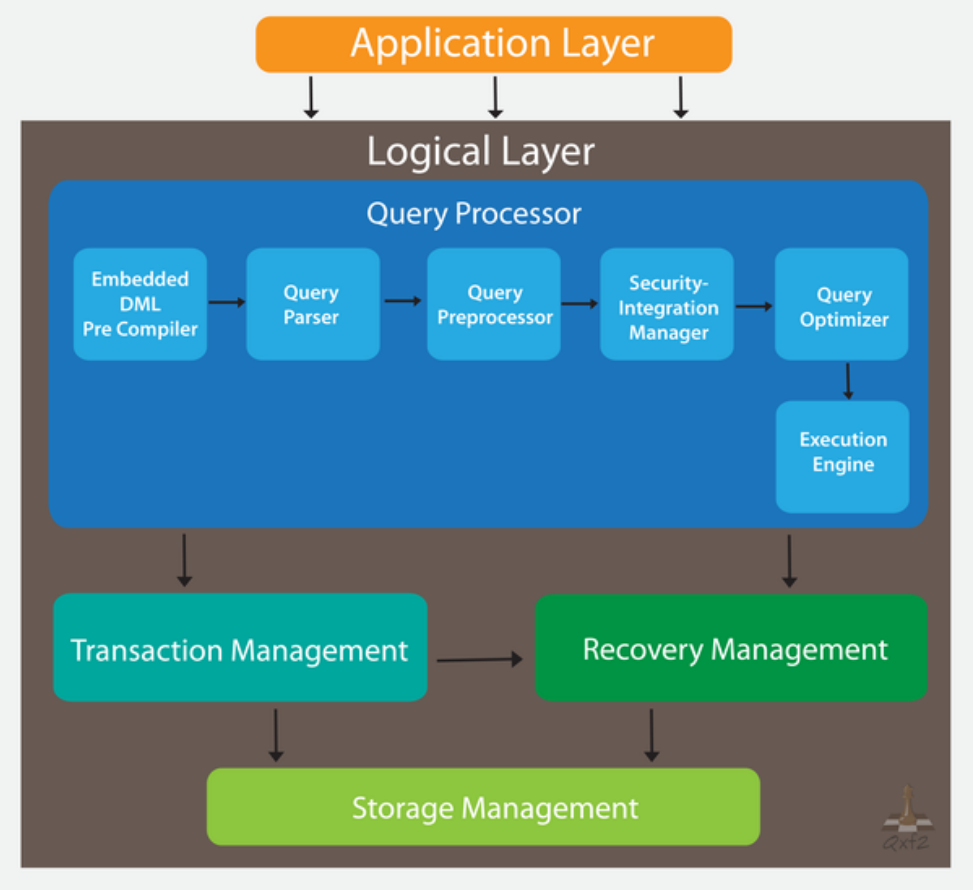


Slika 3 Aplikacijski sloj [6]

- Administratorska komponenta koristi različita sučelja koja izvode zadatke kao što su gašenje servera, kreiranje ili brisanje baze podataka, kopiranje baze podataka, itd.
- Klijentska komponenta komunicira sa MySQL-om preko više različitih sučelja
- Sučelje upita komunicira sa MySQL-om preko sučelja „mysql“

4.2.2. Logički sloj

Podaci se iz logičkoj sloja prosljeđuju u logički sloj. U ovome se sloju nalaze okidači, pogledi te spremljene procedure, a sloj sadrži podsustave za upravljanje oporavkom, transakcijama, skladištenjem te procesuiranjem upita. Ovi upiti podsustavi rade zajedno kako bi obradili zahtjeve koje izdaje MySQL server.



Slika 4 Logička arhitektura MySQL-a. [6]

- Podsustav za procesuiranje upita vrši prijevod klijentske SQL naredbe u naredbe razumljive kompajleru
- Podsustav za upravljanje transakcijama odgovoran je da se svaka transakcija zabilježi i automatski izvrši. U ovaj podsustav spadaju naredbe COMMIT i ROLLBACK
- Podsustav za oporavak služi za ono što i sam naziv govori – oporavak baze podataka. Ukoliko dođe do greške prilikom izvršavanja nekog upita, ovaj podsustav poništi neuspjeli upit
- Podsustav za upravljanje memorijom ima buffer koji alocira memoriju. Prihvaća zahtjeve za izvršenje, te vraća podatke višem sloju

Da bi bolje razumjeli logički sloj, objasnit ćemo detaljnije.

Kad aplikacijski sloj zaprimi zahtjev od strane klijenta za izvršavanje upita, on se izvršava pomoću MySQL sesije. Zadaća je prve komponente unutar podsustava za upravljanje upitima, da prevede klijenti SQL upit u odgovarajuće SQL izjave.

Kada je klijentska sesija uspostavljena, kreira se MySQL nit (*eng. thread*) za procesuiranje svih naredbi i sesija. Nakon toga, upit se prosljeđuje procesu „mysqld“ koji je pokrenut na serveru. Ovo je proces koji je pokrenut u pozadini i upravlja zahtjevima klijenata. Sastoji se od više niti, što omogućava više sesija.

Nakon što upit prođe kroz ove procese, provjerava se da li taj upit već nije izvršen u memoriji upita. Ukoliko je već izvršen ovaj upit, rezultat se vraća iz memorije kako bi se ubrzao proces. Ukoliko se upit ne nalazi u memoriji, kreće se na sljedeću komponentu „Parser“ koji će stvorit strukturu stabla raščlanjivanja na osnovi upita, tako da ostale komponente mogu „razumjeti“ taj upit. Nakon toga se provjerava sintaksa upita, ukoliko nije validna, potrebno je prekinuti operaciju, no, ukoliko je validna, provjerava se da li korisnik ima potrebnu autorizaciju za određenu tablicu kojoj želi pristupiti. U predzadnjoj komponenti donosi se odluka o planu izvršenja i preuzimanje podataka na što brži način. Komponenta za optimizaciju upita odgovorna je za provjeru koji indeksi postoje u navedenoj tablici i ako ih ima, može li se ti indeksi koristiti za brže preuzimanje podataka.

4.2.3. Fizički sloj

Ovaj sloj sadrži komponente za skladištenje podataka, oni su odgovorni za spremanje i dohvaćanje podataka. Postoje različiti mehanizmi za spremanje podataka, no od verzije 5.5.5, zadani mehanizam za spremanje „InnoDB“. Svaki mehanizam ima drukčije karakteristike, te na osnovi aplikacijskih zahtjeva možemo izabrati odgovarajući. Odabir se vrši prilikom izrade tablice.

MySQL pohranjuje svaku bazu podataka kao poddirektorij svojeg direktorija podataka unutar datotečnog sustava. Svaka baza podataka ima pripadajući direktorij. MySQL sprema model tablice u datoteku sa nastavkom „.frm“, a ime te datoteke je ime tablice. [6]

4.3. MySQL sintaksa

Da bi mogli manipulirati podacima unutar sustava za upravljanje podacima MySQL, potrebno je poznavati njegovu sintaksu. Pomoću ispravno napisanog upita možemo doći do željenog rezultata, a da bi malo detaljnije objasnili, prikazat ćemo na primjeru SELECT upita.

Unutar komandne linije možemo unijeti naredbu HELP SELECT koja će nam prikazati moguće opcije ove naredbe:

```
mysql> HELP SELECT;
Name: 'SELECT'
Syntax:
SELECT
    [ALL | DISTINCT | DISTINCTROW ]
      [HIGH_PRIORITY]
      [STRAIGHT_JOIN]
      [SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
      [SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]
    select_expr [, select_expr ...]
    [FROM table_references
      [PARTITION partition_list]
    [WHERE where_condition]
    [GROUP BY {col_name | expr | position}
      [ASC | DESC], ... [WITH ROLLUP]]
    [HAVING where_condition]
    [ORDER BY {col_name | expr | position}
      [ASC | DESC], ...]
    [LIMIT {[offset,] row_count | row_count OFFSET offset}]
    [PROCEDURE procedure_name(argument_list)]
    [INTO OUTFILE 'file_name'
        [CHARACTER SET charset_name]
        export_options
      | INTO DUMPFILE 'file_name'
      | INTO var_name [, var_name]]
    [FOR UPDATE | LOCK IN SHARE MODE]]
```

Slika 5 Prikaz opcija SELECT naredbe

Riječi koje su napisane velikim tiskanim slovom označavaju ključne riječi SQL jezika. Iako su prikazane velikim tiskanim slovima, to ne znači da ih ne možemo pisati malim slovima, no bitno je pisati ih točnim redoslijedom kako je i navedeno u dokumentaciji. Bitno je napomenuti kako sve ključne riječi nisu obavezane prilikom pisanja upita. Ukoliko upit nije ispravno napisan, sustav će nam javiti grešku koju brzo možemo popraviti.[7]

Kako je napomenuto, nisu sve naredbe obavezne. Opcionalne su one koje su napisane unutar uglatih zagrada, no ukoliko ih ima više, te su odvojene okomitom crtom, to znači da možemo koristiti samo jednu od njih. Naredbe koje se nalaze unutar vitičastih zagrada su obavezne naredbe, te ukoliko ih ne unesemo sustav će javiti pogrešku. Nadalje, riječi napisane malim slovima su riječi koje je potrebno zamijeniti nekim nazivom, objektom ili specifičnim opcijama.[7]

Primjer SELECT naredbe:

```
SELECT ime, prezime FROM korisnik;
```

Ovo je najjednostavniji primjer SELECT naredbe. Pošto smo naveli kako želimo samo ime i prezime dohvatiti, ne dohvaćaju se svi podaci, već samo željeni, te smo time uštedjeli na memoriji, a samim time i na brzini izvršavanja upita.

4.4. Tipovi podataka

Da bi mogli kreirati tablicu, moramo biti upoznati sa tipovima podataka koji su dostupni. Naime, kod kreiranja tablice, uz naziv atributa, moramo navesti tip podatka koji ćemo spremati u taj atribut. Odabirom ispravnog podataka izravno utječemo na potrebnu količinu mjesta za pohranu. Kada kreiramo tablicu, moramo se pridržavati tipova podataka prilikom upisa u tablicu, inače će nam sustav javiti grešku. Jedini tip podataka koji možemo upisati u tablicu sa bilo kojim definiranim tipom podataka je NULL, naravno, ukoliko je dozvoljeno unašati tu vrijednost u tablicu.

MySQL podržava više tipova podataka, a možemo ih podijeliti u 3 kategorije:

1. Numerički tip podataka
2. Znakovni tip podataka
3. Datum i vrijeme

4.4.1. Numerički

Pošto postoje različiti tipovi numeričkih podataka, MySQL ih i više podržava. Naime, numerički tipovi podataka podijeljeni su na cjelobrojne, decimalne s nepomičnim zarezom te decimalne s pomičnim zarezom. Sljedeća slika nam prikazuje koliko određeni cjelobrojni tip podatka zauzima memorije, te koja je njegova maksimalna, odnosno minimalna vrijednost.

Type	Storage (Bytes)	Minimum Value Signed	Minimum Value Unsigned	Maximum Value Signed	Maximum Value Unsigned
TINYINT	1	-128	0	127	255
SMALLINT	2	-32768	0	32767	65535
MEDIUMINT	3	-8388608	0	8388607	16777215
INT	4	-2147483648	0	2147483647	4294967295
BIGINT	8	-2^{63}	0	$2^{63}-1$	$2^{64}-1$

Slika 6 Prikaz cjelobrojnih numeričkih tipova podataka u MySQL-u [4]

Kao što je vidljivo iz prethodne tablice, MySQL podržava više tipova cjelobrojnih numeričkih tipova podataka, jedina razlika je u veličini, pa je prilikom kreiranja tablice potrebno odabrati odgovarajući tip podatka.

Decimalne tipove podataka koristimo kada želimo spremiti precizne brojčane vrijednosti nekog podatka, kao što su geografske koordinate.

4.4.2. Znakovni

Kao što i za numerički tip podataka, MySQL podržava više tipova znakovnih podataka. Znakovni tip podataka ima prednost u odnosu na brojčani tip podataka, a to je što osim samih znakova, može pohraniti i brojčani tip. Sljedeća slika prikazuje veličinu pojedinih znakovnih tipa podataka.

Tip	Veličina (znakova)
CHAR	0-255
VARCHAR	0-65536
TINYTEXT	0-255
TEXT	0-65536
MEDIUMTEXT	0-16777215
LONGTEXT	0-4294967295

Slika 7 Prikaz veličine znakovnih tipova podataka

Najčešće korišteni tip podataka je VARCHAR, dok iza njega slijedi CHAR i TEXT. Kao poseban tip potrebno je izdvojiti CHAR koji radi na specifičan način. Naime, ukoliko korisnik unese manje znakova, MySQL sustav će ostali prostor nadopuniti specijalnim znakovima. Dok u suprotnom scenariju, ukoliko korisnik unese previše znakova, sustav će maknuti suvišne znakove. Uz CHAR, specifičan je i VARCHAR koji radi na sličan princip, no razlika je u tome što ukoliko korisnik unese manje znakova, sprema se točno taj niz znakova, bez dodavanja specijalnih. [4]

4.4.3. Datum i vrijeme

MySQL podržava više tipova podataka za spremanje vremena, a to su: DATE, TIME, DATETIME, TIMESTAMP i YEAR. Sljedeća tablica prikazuje zadane vrijednosti, odnosno format spremanja podataka.

Tip podatka	Zadana vrijednost
DATE	'0000-00-00'
TIME	'00:00:00'
DATETIME	'0000-00-00 00:00:00'
TIMESTAMP	'0000-00-00 00:00:00'
YEAR	0000

Tablica 3 Prikaz tip podataka datum i vrijeme

Tip podatka DATE koristi se samo za spremanje datuma, bez vremenskog dijela, a zapisuje se u formatu „GGGG-MM-DD“. Njegov raspon je „1000-01-01“ – „9999-12-31“.

Tip podatka DATETIME koristi se za spremanje datuma zajedno sa vremenskim dijelom. Njegov format je „GGGG-MM-DD hh:mm:ss“. Njego raspon je isti kao i kod tipa DATE uz dodatak sata: „1000-01-01 00:00:00“ – „9999-12-31 23:59:59“.

Tip podatka TIMESTAMP je veoma sličan tipu podatke DATETIME, no podaci za ovaj tip podataka uzimaju se prema UTC vremenu, bez obzira na vremensku zonu. Raspon ovog tipa podatke je „1970-01-01 00:00:01“ UTC – „2038-01-19 03:14:07“ UTC.

Tip podataka YEAR služi za zapisivanje godine. Koriste se brojevi u formatu „YYYY“ ili „YY“ a njegov raspon je „-839:59:59“ – „839:59:59“.[4]

4.4.4. JSON

MySQL također podržava JSON tip podataka. Veličina ovog tip podataka u memoriji jednaka je kao i LONGTEXT. JSON tip podataka ima sljedeće prednosti u odnosu na spremanje JSON formata u TEXT tip podatka:

- Automatska validacija JSON formata. Ukoliko format nije validan, sustav javlja korisniku grešku.
- Optimizacija memorije; podaci se konvertiraju u interni format koja dopušta brzo čitanje informacija. Kada server ima potrebu čitati ove podatke, informacija se može dobiti diskretno prema ključu niza (eng. *array*), pa tako nije potrebno čitanje svih vrijednosti podatka

5. Primjena MySQL SUBP-a za razvoj aplikacije

5.1. ER model

5.2. Opis aplikacijske domene

5.3. Funkcionalnosti aplikacije

6. Zaključak

Popis literature

[1] Oracle, “MySQL Customers.” [Na internetu]. Dostupno: http://www.mysql.com/customers/. [pristupano: 03.09.2019].

[2] C. Tonči and M. Buntić, *Uvod u relacijske baze podataka*. 2015.

[3] E. Udžbenici, “Udžbenici,” 2017. .

[4] D. Axmark and M. Wideniues, “MySQL Documentation,” 2019. [Na internetu]. Dostupno: https://dev.mysql.com/doc/refman/8.0/en/introduction.html. [pristupano: 05.09.2019].

[5] A. Dmitrović, “Kakva će biti sudbina MySQL-a?,” 2012. [Na internetu]. Dostupno: https://sysportal.carnet.hr/node/1119.

[6] I. Nellutla, “MySQL Architecture and Layers,” 2017. [Na internetu]. Dostupno: https://qxf2.com/blog/mysql-architecture-and-layers/. [pristupano: 05.09.2019].

[7] K. Rabuzin, *UVOD U SQL*. 2011.

[8] M. Pavlić, *OBLIKOVANJE BAZA PODATAKA*. 2011.

Popis slika

Popis tablica