Usporedba relacijskih (SQL) i nerelacijskih (NoSQL) baza podataka s primjerima u MySQL-u i MongoDB-u

3.5.2019., Zagreb, Nenad Crnko

Sadržaj

Uvod

Osnovne karakteristike relacijskih (SQL) i nerelacijskih (NoSQL) baza podataka i prikaz njihovih razlika na primjerima baza MySQL i MongoDB

Izvođenje osnovnih operacija kao što su čitanje, upisivanje, ažuriranje i brisanje zapisa u bazi podataka

Naprednije operacije poput optimizacije tablica, indeksa, upita i ostalo

Primjeri korištenja uz pomoć nekoliko programskih jezika (PHP, Python i Visual Studio)

Trenutno stanje na tržištu

Kriteriji za vrednovanje popularnosti (https://db-engines.com)

- Broj upita o pojedinom sustavu na nekoliko najpoznatijih tražilica (Google, Bing i Yandex)
- Opće zanimanje za neki sustav na temelju podataka prikupljenih preko alata Google Trends
- Učestalost tehničkih diskusija o nekom proizvodu koje se provode na *web*-adresama https://stackoverflow.com/ i https://dba.stackexchange.com/
- Broj dostupnih IT poslova povezanih s nekim od sustava na adresama za online pretraživanje ponuđenih poslova https://www.indeed.com/ i https://www.simplyhired.com/
- Učestalost pozivanja na pojedini sustav na društvenim mrežama namijenjenim profesionalcima https://www.linkedin.com/ i https://www.upwork.com/
- Broj pojavljivanja u porukama na https://twitter.com/

The DB-Engines Ranking ranks database management systems according to their popularity. The ranking is updated monthly.

Read more about the method of calculating the scores.

trend chart

345 systems in ranking, March 2019

Rank Mar 2019	Rank Feb 2019	Rank Mar 2018	DBMS	Database Model	Score Mar 2019	Score Feb 2019	Score Mar 2018
1.	1.	1.	Oracle	Relational, Multi-model	1279.14	+15.12	-10.47
2.	2.	2.	MySQL	Relational, Multi-model	1198.25	+30.96	-30.62
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	1047.85	+7.79	-56.94
4.	4.	4.	PostgreSQL	Relational, Multi-model	469.81	-3.75	+70.46
5.	5.	5.	MongoDB	Document	401.34	+6.24	+60.82
6.	6.	6.	IBM Db2	Relational, Multi-model	177.20	-2.23	-9.47
7.	↑ 9.	7.	Microsoft Access	Relational	146.20	+2.18	+14.26
8.	↓ 7.	8.	Redis	Key-value, Multi-model	146.12	-3.32	+14.90
9.	↓ 8.	9.	Elasticsearch	Search engine, Multi-model	142.79	-2.46	+14.25
10.	10.	↑ 11.	SQLite	Relational	124.87	-1.29	+10.06
11.	11.	↓ 10.	Cassandra	Wide column	122.80	-0.58	-0.69
12.	12.	↑ 15.	MariaDB	Relational, Multi-model	84.31	+0.89	+21.21
13.	13.	13.	Splunk	Search engine	83.10	+0.29	+17.44
14.	14.	↓ 12.	Teradata	Relational	75.22	-0.75	+2.76
15.	15.	↑ 18.	Hive	Relational	73.00	+0.71	+16.00
16.	16.	↓ 14.	Solr	Search engine	60.01	-0.95	-4.80
17.	17.	17.	HBase	Wide column	58.80	-1.48	-2.14
18.	18.	↑ 19.	FileMaker	Relational	58.13	+0.34	+3.00
19.	↑ 20.	↓ 16.	SAP Adaptive Server	Relational	56.03	+0.29	-6.58
20.	↓ 19.	20.	SAP HANA	Relational, Multi-model	55.51	-1.03	+6.99
21.	21.	21.	Amazon DynamoDB	Multi-model	54.49	-0.45	+12.04
22.	22.	22.	Neo4j	Graph	48.58	+0.72	+7.68
23.	23.	23.	Couchbase	Document	33.80	-1.78	+0.90
24.	24.	24.	Memcached	Key-value	28.73	-0.72	-2.62
25.	25.	↑ 26.	Microsoft Azure SQL Database	Relational, Multi-model	27.93	+0.81	+3.31
26.	26.	↓ 25.	Informix	Relational, Multi-model	26.90	+0.54	-0.61
27.	27.	↑ 31.	Microsoft Azure Cosmos DB	Multi-model	24.83	-0.03	+8.07
28.	28.	↓ 27.	Vertica	Relational, Multi-model	22.07	-0.74	+1.77
29.	29.	↑ 32.	Amazon Redshift	Relational	20.89	-0.10	+6.70
30.	↑ 32.	↓ 29.	Firebird	Relational	20.17	+0.82	+2.14

Usporedba relacijskih (SQL) i nerelacijskih (NoSQL) baza podataka
(brzi pregled)

Osnovne karakteristike relacijskih (SQL) i nerelacijskih (NoSQL) baza podataka

Relacijske baze - Edgar F. Cood: ***A Relational Model of Data for Large Shared Data Banks*** (*Communications of the ACM, 1970*). Na temelju rada u istraživačkom centru *San Jose* IBM stručnjaci razvili su sustav *System R*.

U *Systemu R* sve operacije su se izvodile pomoću posebnog jezika **Structured English Query Language** (**SEQUEL**). Skraćenica **SEQUEL** u tom trenutku je bila zaštićena od strane tvrtke Hawker-Siddeley.

Naziv skraćen na **SQL** (**Structured Query Language**). Za SQL su zaslužna druga dvojica IBM-ovih stručnjaka (Donald D. Chamberlin i Raymond F. Boyce).

Prvi komercijalni sustavi pojavili su se 1979 godine. IBM je predstavio SQL/DS za računalo System/38. Iste godine Oracle Corporation predstavio je svoj sustav za upravljanje relacijskim bazama podataka (Oracle Version 2).

Neke od ključnih prednosti relacijski orijentiranih baza podataka

- Dobro su poznate u teorijskom i u praktičnom smislu, što rezultira kvalitetnim i stabilnim softverskim rješenjima većeg broja proizvođača.
- Za pristup i izvođenje operacija nad podacima koristi se standardizirani i dobro poznati jezik upita SQL. Za navedeni jezik dostupna je podrška u različitim oblicima (knjige, priručnici, tečajevi i ostalo).
- Primjena relacijski orijentiranih baza podataka dobro je poznata i s financijske strane pa je jednostavnije planirati troškove uvođenja novih sustava ili nadogradnje postojećih.
- Osnovni pojmovi u relacijskom modelu (poput tablica, stupaca i ostalo) jednostavni su za razumijevanje.

Neki od ključnih nedostataka relacijski orijentiranih baza podataka

- Relacijske baze podataka (starije verzije) ne podržavaju neke od oblika podataka koji se sve češće pojavljuju u modernim poslovnim i/ili multimedijalno orijentiranim IT sustavima.
- Kod grupiranja osnovnih vrsta podataka u složenije strukture relacijski model podataka često nije optimalan u pogledu performansi sustava.
- Standardni jezik upita (SQL) također ne predstavlja optimalan izbor za nestandardne oblike podataka.
- Za pristup podacima potrebno je razumjeti strukturu i organizaciju baze podataka.
- Interni mehanizmi zaključavanja resursa nisu pogodni za određene složene vrste podataka.

Najpoznatije vrste NoSQL baza podataka

- **Document**
- Graph
- Columnar Databases
- In-Memory Data Grids

Document baze podataka

- Jedan od najpoznatijih oblika NoSQL baza podataka su **baze dokumenata**. Osnovnu „jedinicu podataka“ predstavlja dokument zajedno s dodatnim metapodacima o dokumentu. Spremanje točno određene vrste dokumenta može se simulirati relacijskim bazama podataka dodatnim stupcima s metapodacima o dokumentu. Problemi nastaju kad u istoj bazi treba čuvati veliku količinu međusobno različitih dokumenata.
- Jedan od najpoznatijih primjera baze dokumenata je MongoDB. Prva verzija razvijena je 2007. godine (tvrtkaMongoDB Inc). MongoDB se za spremanje dokumenata koristi podacima pripremljenim u JSON/BSON stilu uz uključenu podršku za dinamičke sheme podataka. Na početku rada nije potrebno znati sve detalje o tome kakvi dokumenti se spremaju u bazu.

Graph baze podataka

- Temelje se na teoriji grafova, a ključni pojmovi su: **čvorovi i svojstva** (*nodes* i *properties*). Takav oblik strukture podataka pogodan je za uspostavljanje složenih veza među podacima.
- Najpoznatiji primjer sustava iz ove grupe je Neo4j. Riječ je o proizvodu istoimene tvrtke *Neo4j, Inc. (2010)*.
- Dostupna je u dvije verzije – komercijalnoj i besplatnoj – takozvana verzija *Community Edition*. Razlika između verzija je u dodatnim modulima koji omogućavaju upotrebu klaster konfiguracija servera, nadzor nad radom baze podataka i *hot backup*.
- Primjer korištenja demonstriran je pomoću programskog jezika PHP na području s puno međusobnih veza između čvorova – podaci o filmovima i glumcima.

Graph baze podataka

```
$keanu->relateTo($matrix, 'IN')->save();
$laurence->relateTo($matrix, 'IN')->save();
$laurence->relateTo($higherLearning, 'IN')->save();
$jennifer->relateTo($higherLearning, 'IN')->save();
$laurence->relateTo($mysticRiver, 'IN')->save();
$kevin->relateTo($mysticRiver, 'IN')->save();
$path = $keanu->findPathsTo($kevin)
    ->setMaxDepth(12)
    ->getSinglePath();
foreach ($path as $i => $node) {
    if ($i % 2 == 0) {
        echo $node->getProperty('name');
        if ($i+1 != count($path)) {
            echo " was in\n";
        }
    }
}
```

Graph baze podataka

```
$keanu = new Node($client);
$keanu->setProperty('name', 'Keanu Reeves')->save();
$laurence = new Node($client);
$laurence->setProperty('name', 'Laurence Fishburne')->save();
$jennifer = new Node($client);
$jennifer->setProperty('name', 'Jennifer Connelly')->save();
$kevin = new Node($client);
$kevin->setProperty('name', 'Kevin Bacon')->save();

$matrix = new Node($client);
$matrix->setProperty('title', 'The Matrix')->save();
$higherLearning = new Node($client);
$higherLearning->setProperty('title', 'Higher Learning')-
>save();
$mysticRiver = new Node($client);
$mysticRiver->setProperty('title', 'Mystic River')->save();
```

Columnar baze podataka

Naglasak je na brzini pristupa podacima što je veliki problem u velikim bazama podataka. Za pristup podacima i dalje se u pravilu može koristiti SQL kao jezik upita, pa ne treba učiti potpuno nov jezik kao preduvjet za korištenje baze podataka.

Primjer komercijalnog sustava temeljenog na opisanim principima je Amazon Redshift ili MariaDB (posebna konfiguracija).

Kod relacijskog modela baza podataka prethodni podaci spremaju se na disk slijedno u obliku:

```
001:10,Smith,Joe,40000;002:12,Jones,Mary,50000;003:11,Johnson,
Cathy,44000;004:22,Jones,Bob,55000;
```

Za ubrzavanje pristupa podacima mogu se koristiti indeksi. Na primjer:

```
001:40000;002:50000;003:44000;004:55000;
```

Columnar baze podataka

Kod baza podataka orijentiranih prema stupcima, podaci se na disku spremaju u drugačijem redoslijedu u datoteku na disku (ili čak kao više odvojenih datoteka):

```
10:001,12:002,11:003,22:004;Smith:001,Jones:002,Johnson:003,Jones:004;Joe:001,Mary:002,Cathy:003,Bob:004;40000:001,50000:002,44000:003,55000:004;
```

Podaci u ovom modelu izgledaju kao veći broj indeksa iz relacijskog modela navedenih jedan za drugim. U slučaju ponavljanja istog podatka moguće je koristiti sažimanje nabrajanjem svih mjesta na kojima se isti podatak. Na primjer:

```
…;Smith:001,Jones:002,004,Johnson:003;…
```

Operacije čitanja podataka mnogo su brže nego kod relacijskog modela, jer se u većini slučajeva čitaju samo točno određeni podaci. Kod ažuriranja podataka nema potrebe za održavanjem brojnih indeksa kao u relacijskom modelu.

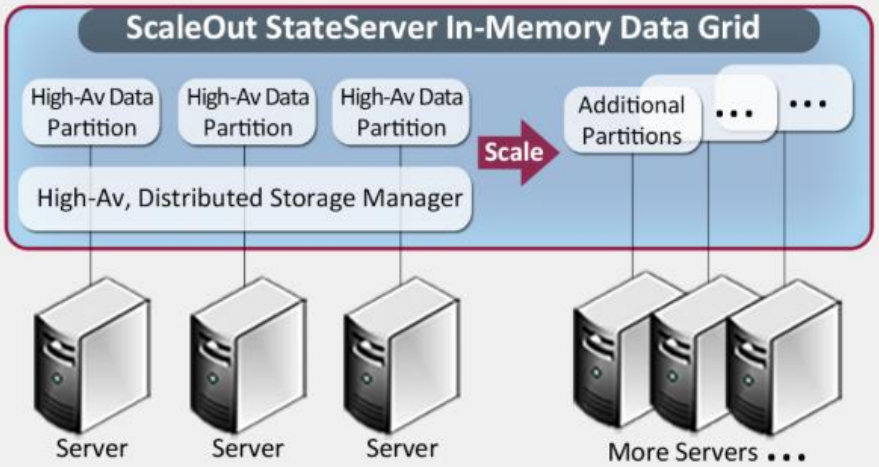
In-Memory Data Grids

In-Memory Data Grids baze ne predstavljaju samo preslikavanje podataka s diskova u memoriju računala, nego izmjenama u organizaciji podataka nude povećanje performansi u radu. Umjesto čuvanja podataka u obliku slogova ili stupaca u ovom modelu podaci se obično čuvaju kao kombinacija vrijednosti ključeva i samih vrijednosti (parovi key/value).

To dovodi do ograničavanja nekih od standardnih mogućnosti relacijskih baza podataka (na primjer, dijela mogućnosti SQL upita ili indeksiranja), ali su osnova za brojne prednosti kod sustava s izrazito velikim brojem procesora (brzina izvođenja upita te mogućnost uključivanja dodatnih hardverskih resursa u sustav).

Kao primjer proizvoda zasnovanog na principima *In-Memory Data Grids* možemo navesti ScaleOut StateServer.

In-Memory Data Grids



Izvor: https://www.scaleoutsoftware.com/products/stateserver/

Omogućava jednostavno uključivanje dodatnih hardverskih resursa. Podrška za standardni i paralelni pristup u različitim programskim jezicima (Java, .NET i C/C++).

MySQL i MariaDB (međusobne razlike)

MySQL i MariaDB - usporedba

MySQL je razvila švedska tvrtka MySQL AB 1995. godine. Sun Microsystems preuzima sustav 2008. godine, a od 2010. u vlasništvu Oraclea (nakon kupovine Sun Microsystems).

Dio originalnih autora MySQL-a (Michael Widenius i ostali) razvio je poseban (kompatibilan) proizvod pod nazivom **MariaDB**.

Trenutno stanje:

345 systems in ranking, March 2019

Rank			DBMS	Database Model	Score		
Mar 2019	Feb 2019	Mar 2018			Mar 2019	Feb 2019	Mar 2018
1.	1.	1.	Oracle	Relational, Multi-model	1279.14	+15.12	-10.47
2.	2.	2.	MySQL	Relational, Multi-model	1198.25	+30.96	-30.62
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	1047.85	+7.79	-56.94
4.	4.	4.	PostgreSQL	Relational, Multi-model	469.81	-3.75	+70.46
5.	5.	5.	MongoDB	Document	401.34	+6.24	+60.82
6.	6.	6.	IBM Db2	Relational, Multi-model	177.20	-2.23	-9.47
7.	↑ 9.	7.	Microsoft Access	Relational	146.20	+2.18	+14.26
8.	↓ 7.	8.	Redis	Key-value, Multi-model	146.12	-3.32	+14.90
9.	↓ 8.	9.	Elasticsearch	Search engine, Multi-model	142.79	-2.46	+14.25
10.	10.	↑ 11.	SQLite	Relational	124.87	-1.29	+10.06
11.	11.	↓ 10.	Cassandra	Wide column	122.80	-0.58	-0.69
12.	12.	↑ 15.	MariaDB	Relational, Multi-model	84.31	+0.89	+21.21
13.	13.	13.	Splunk	Search engine	83.10	+0.29	+17.44
14.	14.	↓ 12.	Teradata	Relational	75.22	-0.75	+2.76

MySQL i MariaDB - razlike

XtraDB Storage Engine
Do verzije 10.2 s boljim performansama od InnoDB. Od 10.2 također InnoDB.

ColumnStore Storage Engine
Dostupan u MariaDB kao pomoć kod rada s vrlo velikim količinama podataka.

MyRocks Storage Engine
Dostupan u MariaDB za rad s kompresiranim podacima (2x i 3-4x bolja kompresija od InnoDB).

Aria Storage Engine
Moderno unapređenje za MyISAM.

MySQL i MariaDB - razlike

Rječnik podataka
Brojne razlike i unapređenja u MySQL u odnosu na MariaDB.

Brzina izvođenja
Bolja optimizacija pogleda kod MariaDB jer se propituju samo stvarno potrebne tablice u pogledima, te paralelnog izvođenja upita.

Replikacija baze podataka
Izvodi se pomoću MySQL Cluster i MariaDB Galera.

MySQL i MariaDB - kompatibilnost

- Na razini strukture baze podataka (tablice, indeksi, ...)
- Na razini interne strukture indeksa
- Protokoli na strani klijenta, strukture, API pozivi
- MySQL Connector moduli rade bez izmjena
- Alati u komandnoj liniji (mysqladmin, mysqldump...)
- Na razini DML upita (select, insert, update, delete)
- Uspoređivanje i ujednačavanje koda na mjesečnoj razini (MariaDB)

MySQL i MariaDB - korisnici

Primjeri korisnika MySQL

GitHub, US Navy, NASA, Tesla, Netflix, WeChat, Facebook, Zendesk, Twitter, Zappos, YouTube...

https://www.mysql.com/customers/

Primjeri korisnika MariaDB

Google, Craigslist, Wikipedia, archlinux, RedHat, CentOS, Fedora...

MySQL i MariaDB – razvojni alati i distribucija

MySQL

Razvojni alati: C i C++

Postoje distribucije za Microsoft Windows, OS X, Linux, AIX, BSDi, FreeBSD, HP-UX, IRIX, NetBSD, Novell Netware, ...

Preuzimanje: https://www.mysql.com/downloads/

MariaDB

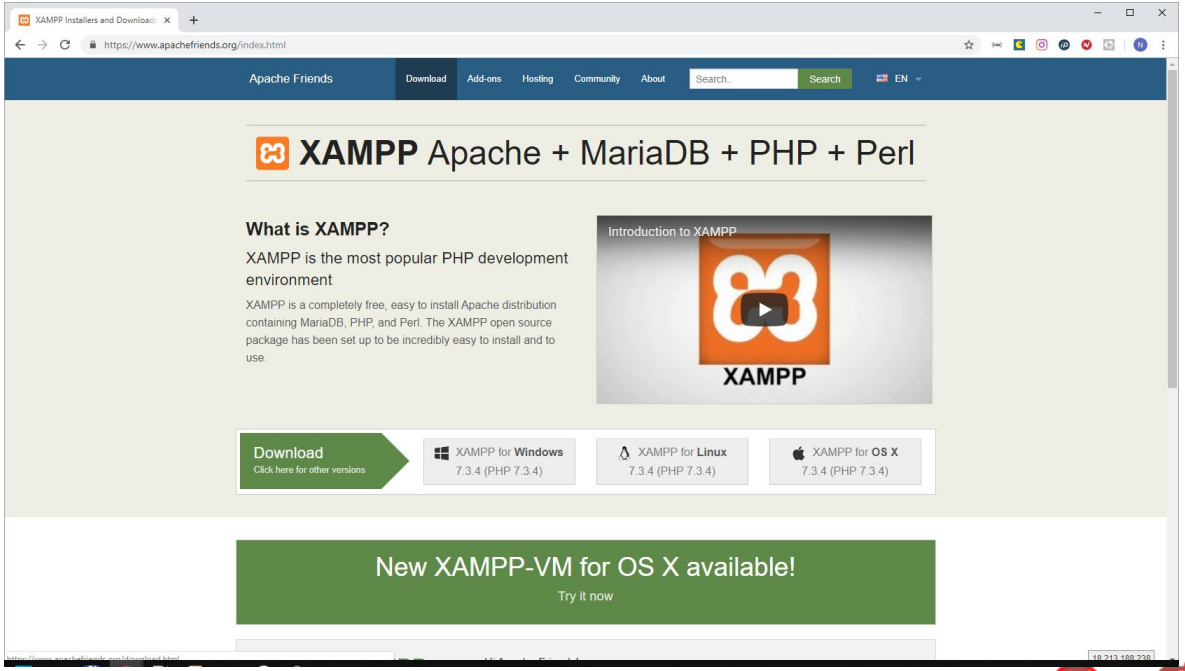
Razvojni alati: C i C++, Bash i Perl

Postoje distribucije za Microsoft Windows, Linux, OS X, FreeBSD, OpenBSD, Solaris...

Preuzimanje: https://downloads.mariadb.org/

MySQL i MariaDB – kao dijelovi razvojnih okruženja

https://www.apachefriends.org/index.html

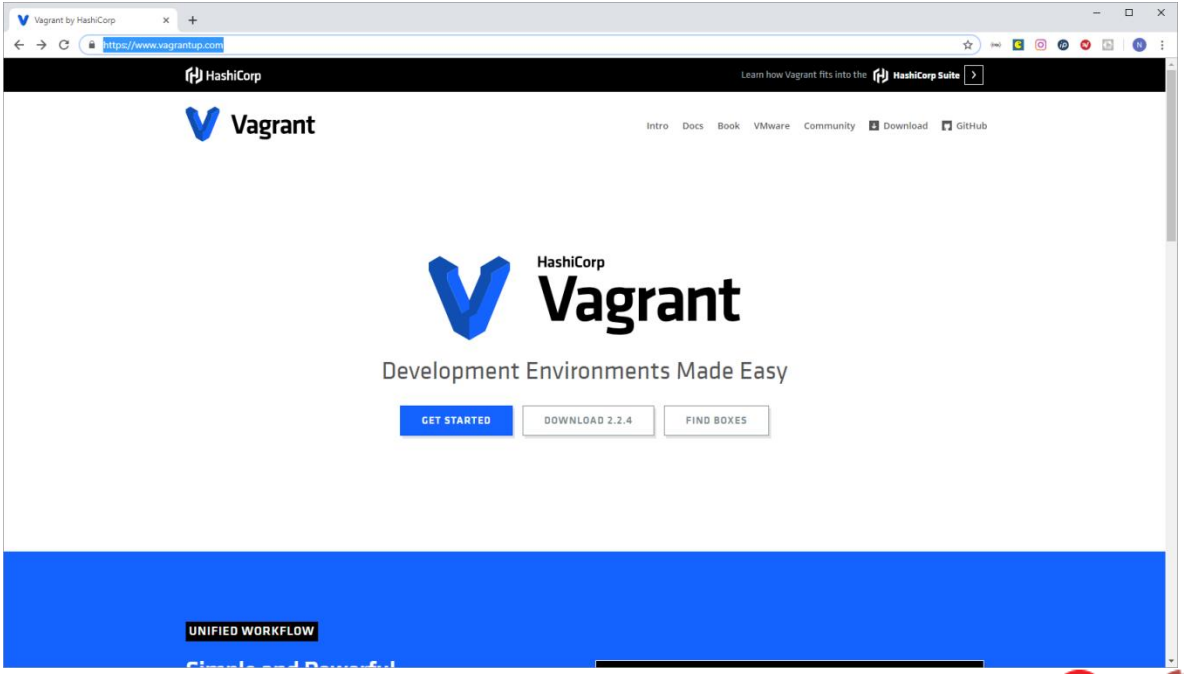


MySQL i MariaDB – kao dijelovi razvojnih okruženja

https://www.ampps.com/

MySQL i MariaDB – kao dijelovi razvojnih okruženja

https://www.vagrantup.com/



MySQL i MariaDB – dijelovi razvojnih okruženja

Zamjena baza MySQL s MariaDB u konfiguracijskoj datoteci za Vagrant

```
ip: "192.168.10.10"
memory: 2048
cpus: 1
provider: virtualbox

mariadb: true
mongodb: true

...
```

MySQL i MariaDB – dodatni materijali

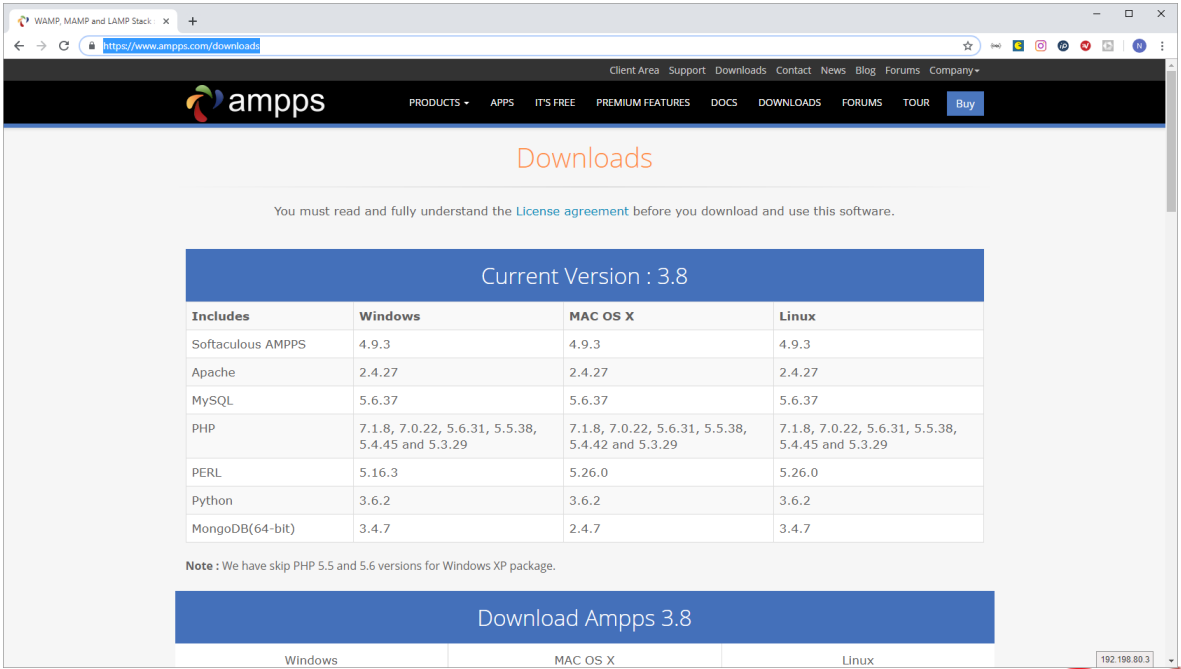
- https://sistemac.srce.hr/seminar-za-it-specijaliste-optimiziranje-mysql-baze-i-laravela-za-rad-s-vrlo-velikim-bazama-podataka

- https://sistemac.srce.hr/mysql-storage-engine-118
- https://sistemac.srce.hr/vrste-indeksa-prema-internoj-strukturi-125
- https://sistemac.srce.hr/koristenje-hash-indeksa-kod-pretrazivanja-mysql-baze-podataka-123
- https://sistemac.srce.hr/mysql-sto-donosi-nova-verzija-121

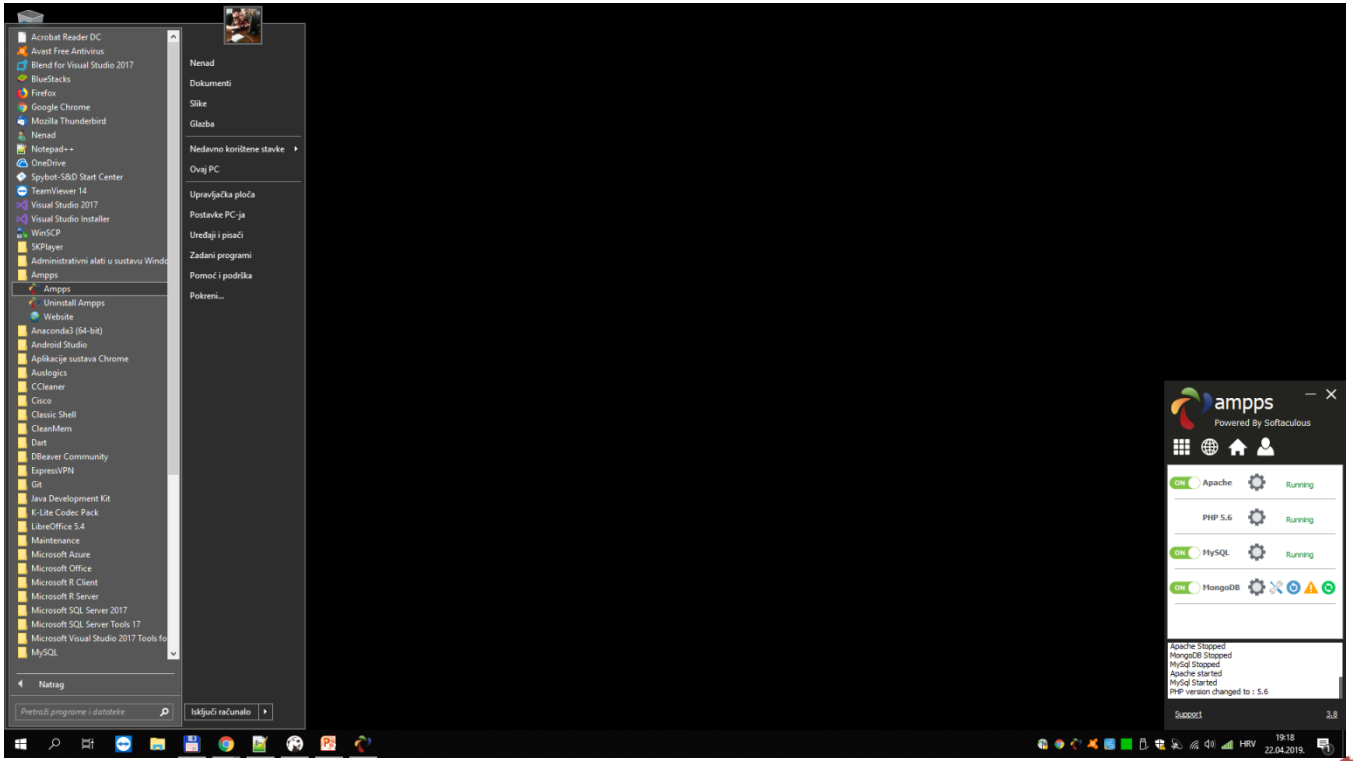
MySQL i MongoDB
Preuzimanje, instalacija i konfiguracija

MySQL i MongoDB – instalacija

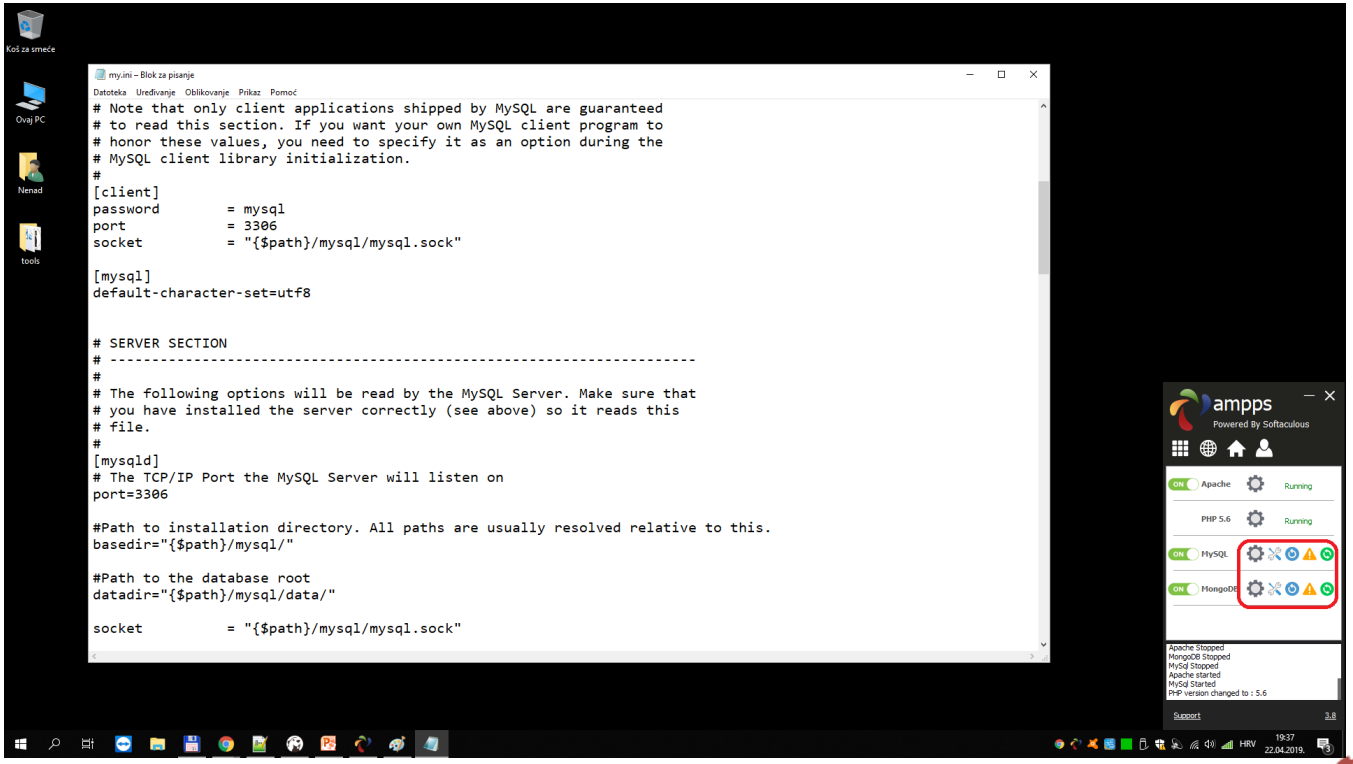
https://www.ampps.com/downloads (Windows instalacija)



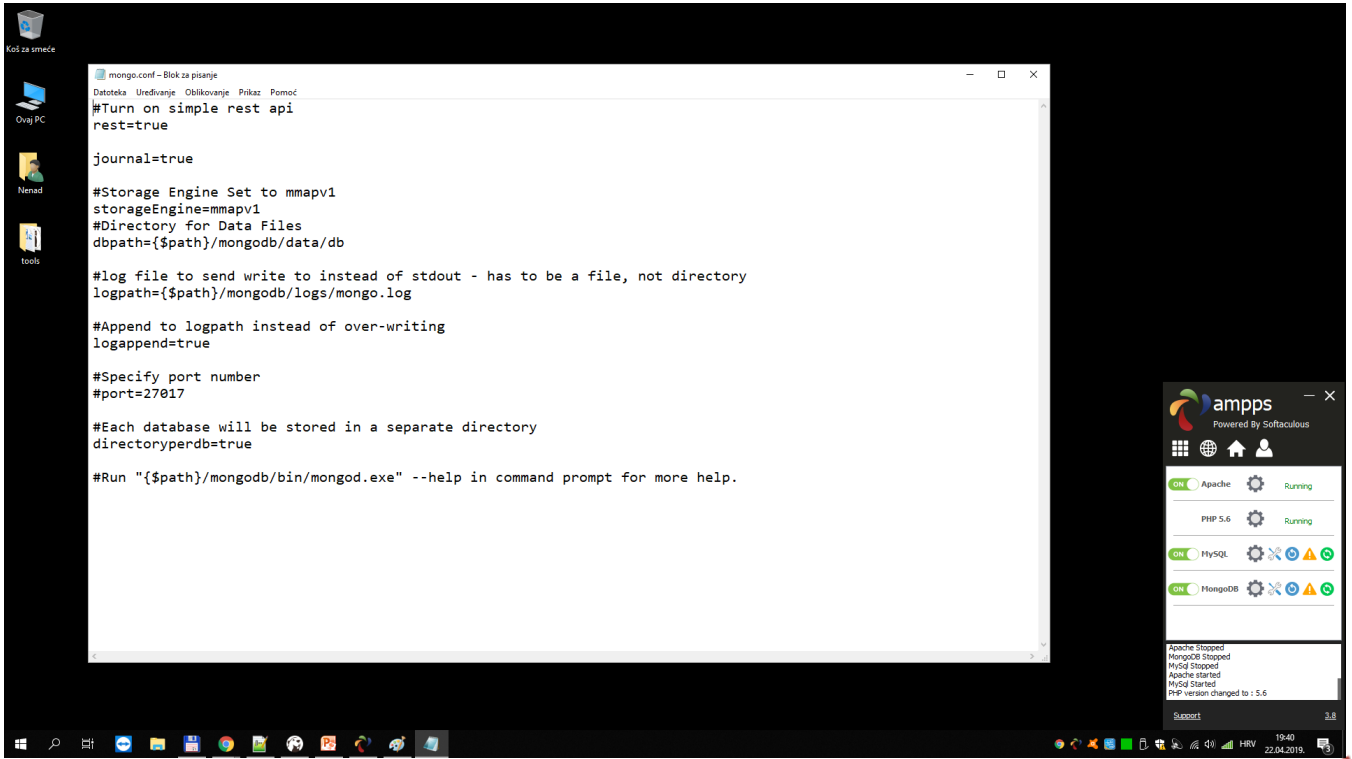
MySQL i MongoDB – pokretanje



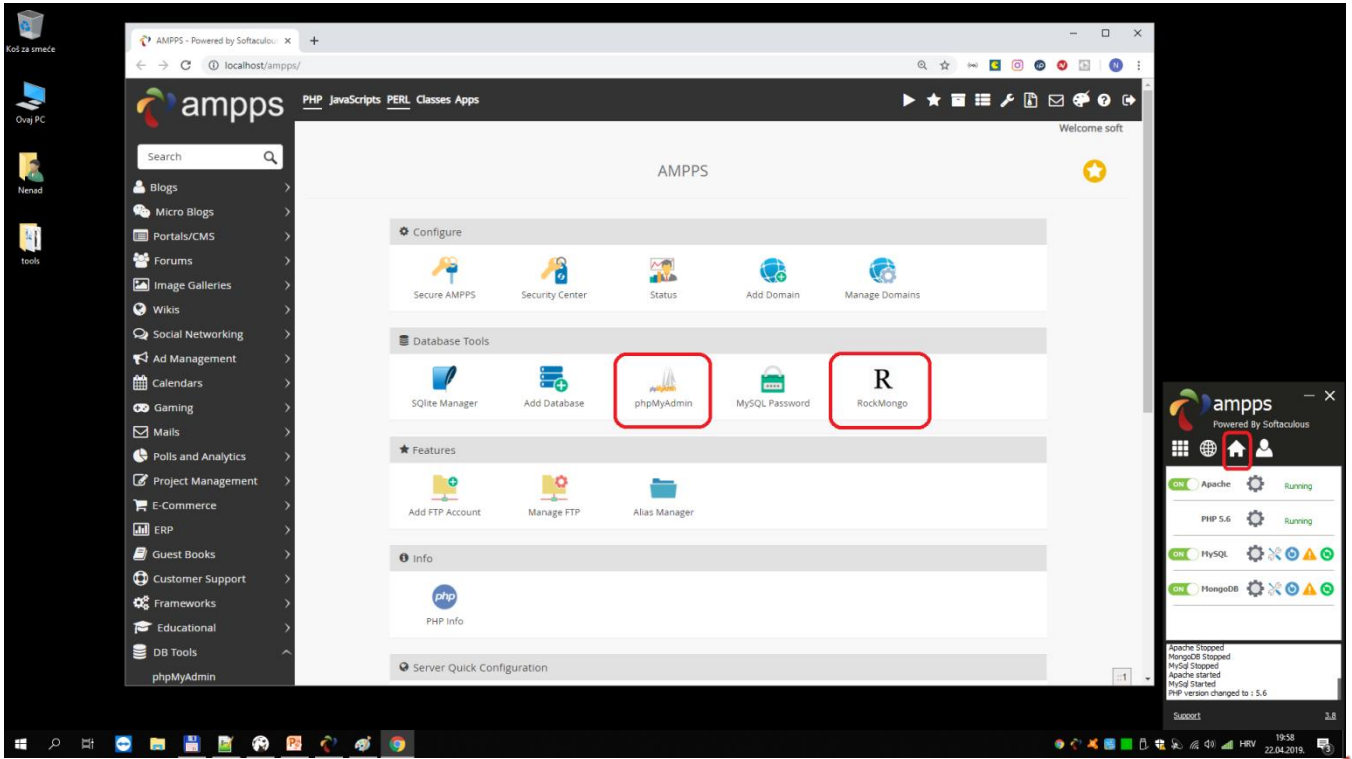
MySQL i MongoDB – konfiguracija



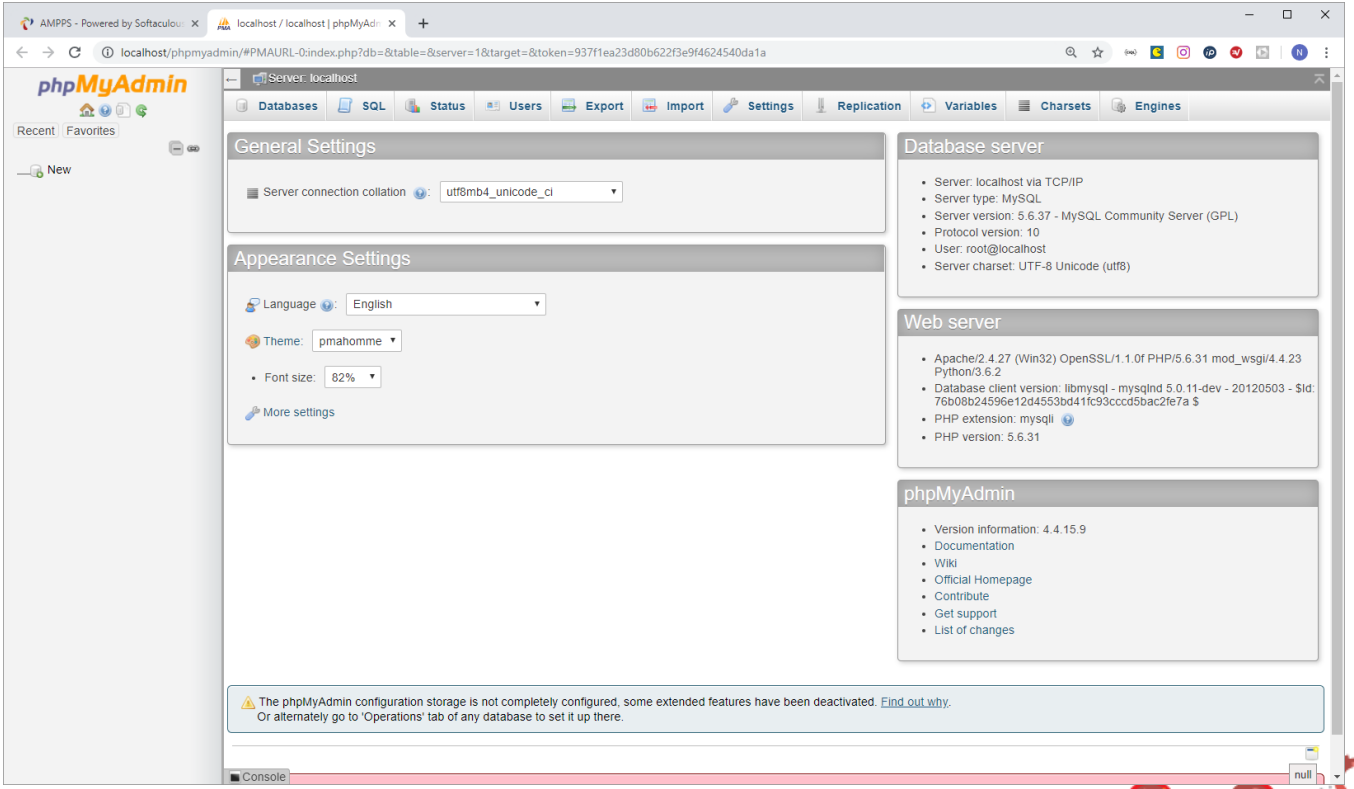
MySQL i MongoDB – konfiguracija



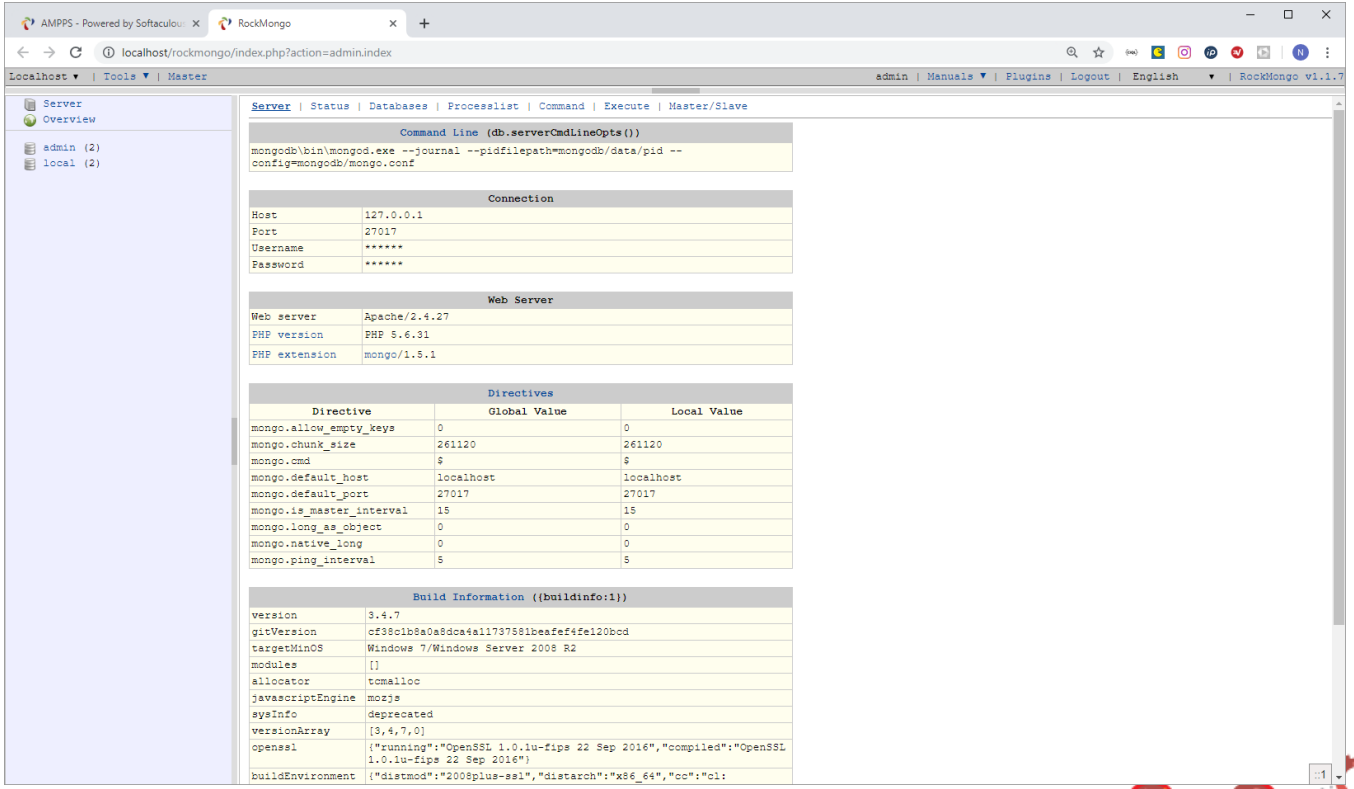
MySQL i MongoDB – korisničko sučelje



MySQL - phpMyAdmin



MongoDB - RockMongo



Alternativna korisnička sučelja za MySQL

MySQL Workbench
https://www.mysql.com/products/workbench/

HeidiSQL
https://www.heidisql.com/

dbForge Studio for MySQL
https://www.devart.com/dbforge/mysql/studio/

Navicat for MySQL
https://www.navicat.com/en/products/navicat-for-mysql

Dbeaver
https://dbeaver.io/

Alternativna korisnička sučelja za MongoDB

MongoDB Compass & Ops Manager
https://www.mongodb.com/products/compass

noSQLBooster
https://nosqlbooster.com/downloads

Studio 3T
https://studio3t.com

Navicat for MongoDB
https://navicat.com/en/products/navicat-for-mongodb

Dbeaver
https://dbeaver.io/

MySQL i MongoDB
Osnovni pojmovi i operacije nad podacima

MySQL i MongoDB – organizacija podataka

MySQL

Podaci se u bazi podataka spremaju u obliku tablica koje moraju biti definirane prije nego se podaci mogu zapisati u bazu podataka. Kako bi se izbjegla redundancija i druge anomalije u podacima nad tablicama se prije pripreme provodi normalizacija. Promjena modela baze može biti značajna u slučaju pojave novog podatka.

MongoDB

Podaci se u bazu podataka spremaju u BSON (Binary JSON) formatu za koji nije potrebno unaprijed pripremiti shemu za spremanje. U istu kolekciju dokumenata mogu se spremiti različite strukture dokumenata.

MySQL – osnovni pojmovi

TABLICA

Dvodimenzionalni objekt u relacijskoj bazi podataka sastavljen od redova i stupaca namijenjen za spremanje podataka. U svaku od tablica spremaju se podaci o jednom od objekata kreiranih tijekom faze dizajniranja baze podataka. Nekoliko mogućih primjera tablica su: učenici, škole, nastavnici, pošte, uplate itd. Stupci tablice predstavljaju atribute objekta (npr. prezime, ime, jmbg, naselje), a svaki red tablice predstavlja jedno pojavljivanje objekta.

Baza podataka se može sastojati od samo jedne ili od samo nekoliko tablica, dok se kod složenih IT sustava njihov broj može popeti do razine od više stotina entiteta.

MySQL – osnovni pojmovi

NULL VRIJEDNOST

Podatak zapisan u tablicu bez točno definirane vrijednosti. Null vrijednost se razlikuje od podatka nula ili praznog niza znakova pa se ne smije koristiti umjesto njih. Prema ANSI standardima Null vrijednost ne bi trebalo koristiti u operacijama uspoređivanja, a ni u računskim operacijama s drugim vrijednostima (uključujući drugu Null vrijednost).

PODRAZUMIJEVANA VRIJEDNOST

Podrazumijevana vrijednost predstavlja zamjenska vrijednost definiranu na razini stupca tablice. U slučaju da se kod dodavanja sloga ne navede vrijednost stupca na to mjesto se automatski upisuje podrazumijevana vrijednost.

Podrazumijevane vrijednosti bi trebalo koristiti u slučajevima kad su određene vrijednosti podataka u tablicama bitno češće od drugih vrijednosti.

MySQL – osnovni pojmovi

INDEKS

Posebna vrsta objekata u relacijskoj bazi podataka namijenjen za ubrzavanje pristupa podacima u pripadajućoj tablici.

Dodatna mogućnost u korištenju indeksa je sprečavanje upisa ponavljajućih vrijednosti u određeni stupac tablice - takozvani jedinstveni (**unique**) **indeks**.

Dodavanje svakog novog indeksa izaziva dodatno zauzeće prostora u bazi podataka, što kod tablica reda veličine više milijuna slogova može predstavljati prilično opterećenje prostora. Još važniji razlog za kontrolu broja indeksa je usporavanje operacija ažuriranja podataka u tablici, jer se kod svake takve operacije mora izvesti ažuriranje pripadajućih indeksa.

(Napomena: Indeksi postoje i u MongoDB!)

MySQL – osnovni pojmovi

OGRANIČENJE

Predstavlja dodatno svojstvo određenog stupca tablice. Pravilnim definiranjem ograničenja baza podataka može samostalno (bez posebne provjere u aplikativnom softveru) filtrirati i spriječiti upis neispravnih podataka.

Problem je što svako dodatno ograničenje utječe na performance sustava čak i ako nema mogućnosti da bude narušeno.

(Napomena: Posebna vrsta ograničenja postoje i u MongoDB!)

MySQL – osnovni pojmovi

VANJSKI KLJUČ

Jedan ili više stupaca tablice čije se vrijednosti podudaraju s primarnim ključem iste ili druge tablice. Vanjski ključ tablice koristi se za kreiranje veze prema primarnom ključu druge tablice. Nakon kreiranja takve veze baza podataka automatski brine o održavanju integriteta podataka.

Na primjer, u stupce vanjskog ključa prva tablice nije moguće upisati vrijednosti koje ne postoje u primarnom ključu druge tablice, a iz druge tablice nije dozvoljeno brisanje vrijednosti primarnog ključa već upotrijebljene u prvoj tablici.

Također mogućnost koja utječe na brzinu rada sustava.

MySQL – osnovni pojmovi

SQL (STRUCTURED QUERY LANGUAGE)

DCL (Data Control Language) – podskup naredbi za određivanje prava nad objektima u bazi podataka. Pomoću naredbi iz ove grupe mogu se odrediti prava pristupa korisnika određenim tablicama ili stupcima tablice (npr. financijski podaci).

DDL (Data Definition Language) - podskup naredbi za definiranje svih atributa i svojstava baze podataka i njezinih objekata (npr. fizičkih datoteka baze podataka, stupaca u tablici i slično).

DML (Data Manipulation Language) - podskup naredbi za izvođenje operacija nad podacima u bazi podataka: pregled, dodavanje, ažuriranje i brisanje podataka (najčešće korištena vrsta).

MySQL – osnovni pojmovi

POHRANJENA PROCEDURA

Logička cjelina sastavljena od većeg broja Transact-SQL naredbi prevedena i spremljena u bazu podataka pod vlastitim nazivom.

Pohranjena procedura se u pravilu izvodi brže od ekvivalentnog broja pojedinačnih SQL naredbi, jer ne treba gubiti vrijeme na njezinu analizu i prevođenje. Znatno je smanjen i mrežni promet između aplikacije na strani klijenta i pohranjene procedure, jer je umjesto niza pojedinačnih SQL naredbi dovoljno poslati naziv procedure i pripadajuće parametre.

Omogućava razvoj aplikacija i manje iskusnim programerima.

MySQL – osnovni pojmovi

OKIDAČ

Posebna vrsta pohranjene procedure koja se automatski izvodi nakon (ili umjesto) određene promjene podataka u odgovarajućoj tablici. To znači da se okidač izvodi nakon svake promjene podataka u pripadajućoj tablici bez obzira na alat u kojem je promjena napravljena (vlastita aplikacija, grafičko korisničko sučelje baze podataka, ili neki od nezavisnih alata).

Na taj način postiže se puno veća pouzdanost u izvođenju kritičnih operacija, nego u slučaju ručnog pokretanja ekvivalentnih procedura.

Treba paziti na međusobno rekurzivno pozivanje okidača.

MySQL – osnovni pojmovi

NORMALIZACIJA

Postupak izmjene structure i sadržaja tablica u bazi podataka, s ciljem uklanjanja redundancije podataka u bazi te dobivanja modela podataka s jednostavnim održavanjem integriteta baze podataka.

DENORMALIZACIJA

Postupak suprotan od prethodnog. Zbog bržeg izvođenja nekih operacija nad bazom podataka (najčešće čitanja podataka iz tablica) u tablice se svjesno dodaje djelomična redudancija u podacima.

NORMALIZACIJA TABLICA

Primjer početnih podataka i prateći problemi:

Nadređeni	Podređeni 1	Podređeni 2	Podređeni 3	Podređeni 4
Branko	Josip	Marija	Stanko	
Marija	Mihaela	Slaven	Ksenija	Denis
Josip	Alan			

Nije moguće dodati novog podređenog.
Neracionalno korištenje prostora za mali broj podređenih.

Međurješenje (problem pojave istog podatka) i rješenje

Nadređeni	Podređeni
Branko	Josip
Branko	Marija
Branko	Stanko
Marija	Mihaela
Marija	Slaven
Marija	Ksenija
Marija	Denis
Josip	Alan

Nadređeni	Podređeni
1	3
1	2
1	4
2	5
2	6
2	7
2	8
3	9

DRUGA NORMALNA FORMA

Pomoću druge normalne forme želi se postići uklanjanje redundantnih podataka iz tablice njihovim izdvajanjem u nove tablice te uspostavljanjem odgovarajućih veza između polazne tablice i novih tablica.

Šifra	Prezime	Ime	Adresa	Broj pošte	Naziv pošte
1	Mernjak	Mihaela	Ognjena Price 26	42000	Zagreb
2	Luka	Horvat	Ognjena Price 28	42000	Varaždin
3	Marija	Gregurić	Slavonska 6	10000	Zagreb
4	Stanko	Lovrić	Braće Radić 24	42000	Varaždin

Nepotrebno se troši prostor za spremanje naziva pošte u tablici.

Otežano je ažuriranje tablice. U slučaju da radnik promijeni mjesto stanovanja (a samim tim i poštu) osim broja pošte, u tablici treba ažurirati i njezin naziv. Još gora situacija nastaje u slučaju promjene naziva pošte. Novi naziv pošte treba upisati svim radnicima u tablici.

DRUGA NORMALNA FORMA

Rješenje uz spajanje vanjskog ključa jedne tablice s primarnim ključem druge tablice.

Šifra	Prezime	Ime	Adresa	Broj pošte
1	Mernjak	Mihaela	Ognjena Price 26	42000
2	Luka	Horvat	Ognjena Price 28	42000
3	Marija	Gregurić	Slavonska 6	10000
4	Stanko	Lovrić	Braće Radić 24	10000

Broj pošte	Naziv
10000	Zagreb
42000	Varaždin

TREĆA NORMALNA FORMA

Trebaju biti zadovoljeni svi uvjeti za prvu i drugu normalnu formu.

Iz tablice treba ukloniti stupce koje nisu u potpunosti ovisne o primarnom ključu.

Broj narudžbe	Šifra kupca	Cijena	Količina	Iznos
1	12	700	2	1400
2	57	100	2	200
3	57	99	1	99
4	123	77	10	770

Prethodna tablica narudžbi već se nalazi u prvoj normalnoj formi jer nema višestrukog ponavljanja istog stupca i postoji stupac za jedinstveno određivanje svakog reda tablice (**Broj narudžbe**). U tablici nema redundantnih podataka pa je zadovoljena i druga normalna forma.

Tablica nije u trećoj normalnoj formi jer stupac Iznos nije u potpunosti ovisan o primarnom ključu – dobiva se jednostavnim množenjem stupaca cijena i količina.

TREĆA NORMALNA FORMA

Rješenje nakon izbacivanja redundatnog reda:

Broj narudžbe	Šifra kupca	Cijena	Količina
1	12	700	2
2	57	100	2
3	57	99	1
4	123	77	10

JSON

JSON (**J**ava**S**cript **O**bject **N**otation)

Samoopisni format jednostavan za razumijevanje namijenjen spremanju i prijenosu podataka.

https://www.json.org/, https://tools.ietf.org/html/rfc7159.html

```
{
"employees":[
    {"firstName":"John", "lastName":"Doe"},
    {"firstName":"Anna", "lastName":"Smith"},
    {"firstName":"Peter", "lastName":"Jones"}
]
}
```

BSON

BSON (**B**inary J**SON**)

Omogućava zapis dodatnih podataka poput datuma ili binarnih podataka što MongoDB koristi kod spremanja dokumenata.

http://bsonspec.org/spec.html

Primjer za {"hello":"world"}

```
\x16\x00\x00\x00                  // total document size
 \x02                             // 0x02 = type String
 hello\x00                        // field name
 \x06\x00\x00\x00world\x00        // field value (size of
                                      value, value, null
                                      terminator)
 \x00                             // 0x00 = type EOO ('end of
                                      object')
```

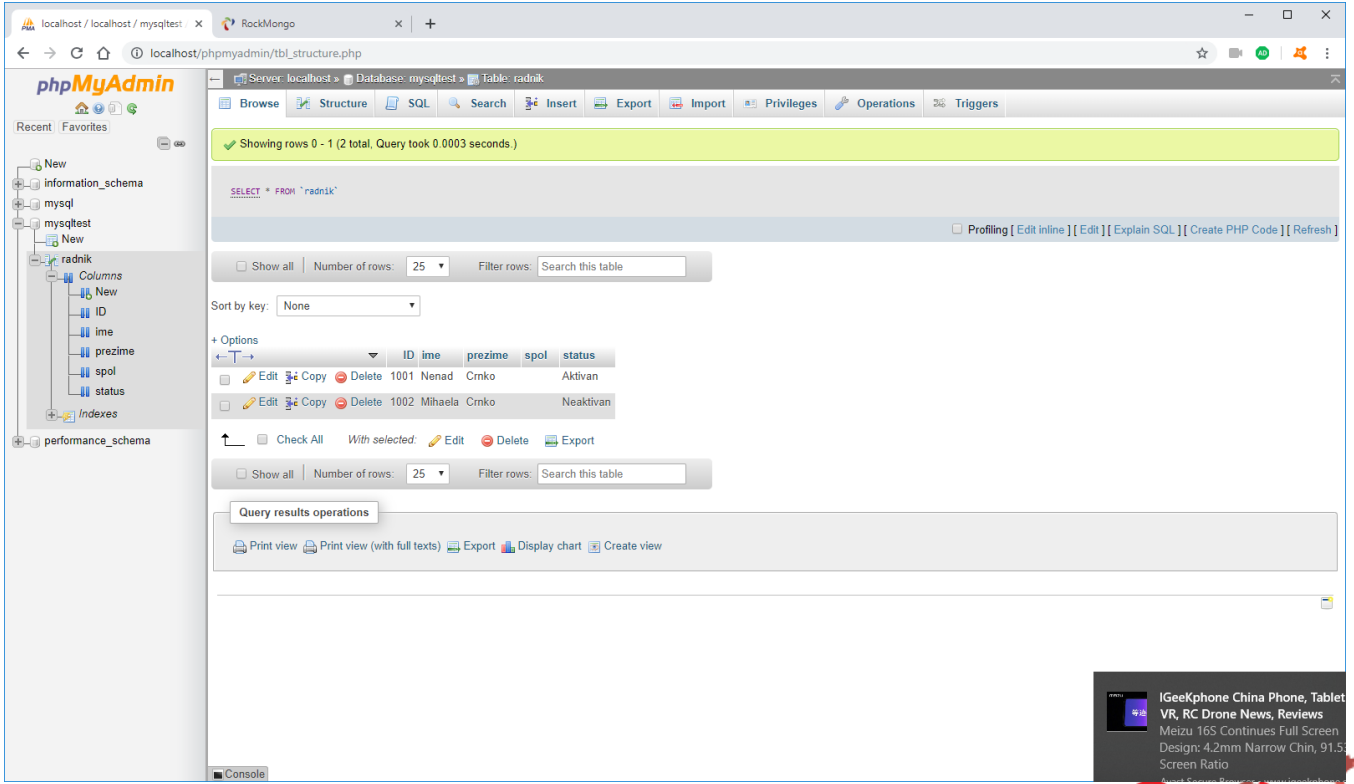
JSON i BSON usporedba

	JSON	BSON
Vrsta zapisa	Standardni format zapisa	Binarni format zapisa
Brzina u radu	Sporiji	Brži
Zauzeće prostora	U pravilu manje zauzeće	U pravilu veće zauzeće
Korištenje	Spremanje i prijenos podataka	Spremanje podataka
Kodiranje i dekodiranje	Nema	Postoji (brzo)
Struktura	Jezično nezavisni format	Elementi sadrže naziv, vrstu i vrijednost
Obuhvat	Analizira se cijeli sadržaj	Samo relevatni sadržaj
Parsing	Nije potreban	Potreban je ali je brz

Primjer rada s podacima - MySQL

```
DROP DATABASE IF EXISTS mysqltest;

CREATE DATABASE mysqltest CHARACTER SET utf8 COLLATE utf8_general_ci;

DROP TABLE IF EXISTS radnik;

CREATE TABLE radnik (
     ID smallint unsigned not null auto_increment,
     ime varchar(30) not null,
     prezime varchar(30) not null,
     spol varchar(1) not null,
     status varchar(15) not null,
     constraint pk_radnik primary key (ID) );


INSERT INTO radnik ( ID, ime, prezime, status ) VALUES ( 1001, 'Nenad',
'Crnko', `Aktivan' );

INSERT INTO radnik ( ID, ime, prezime, status ) VALUES ( 1002, 'Mihaela',
'Crnko', 'Neaktivan' );
```

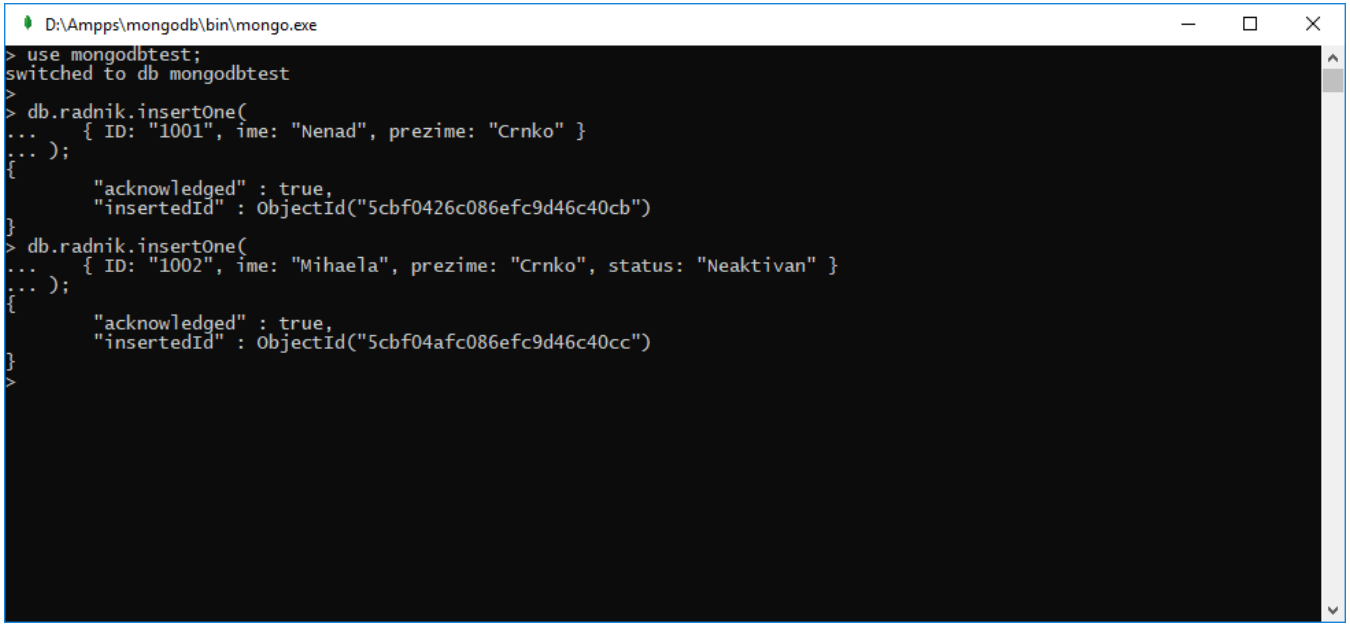
Rezultati izvođenja



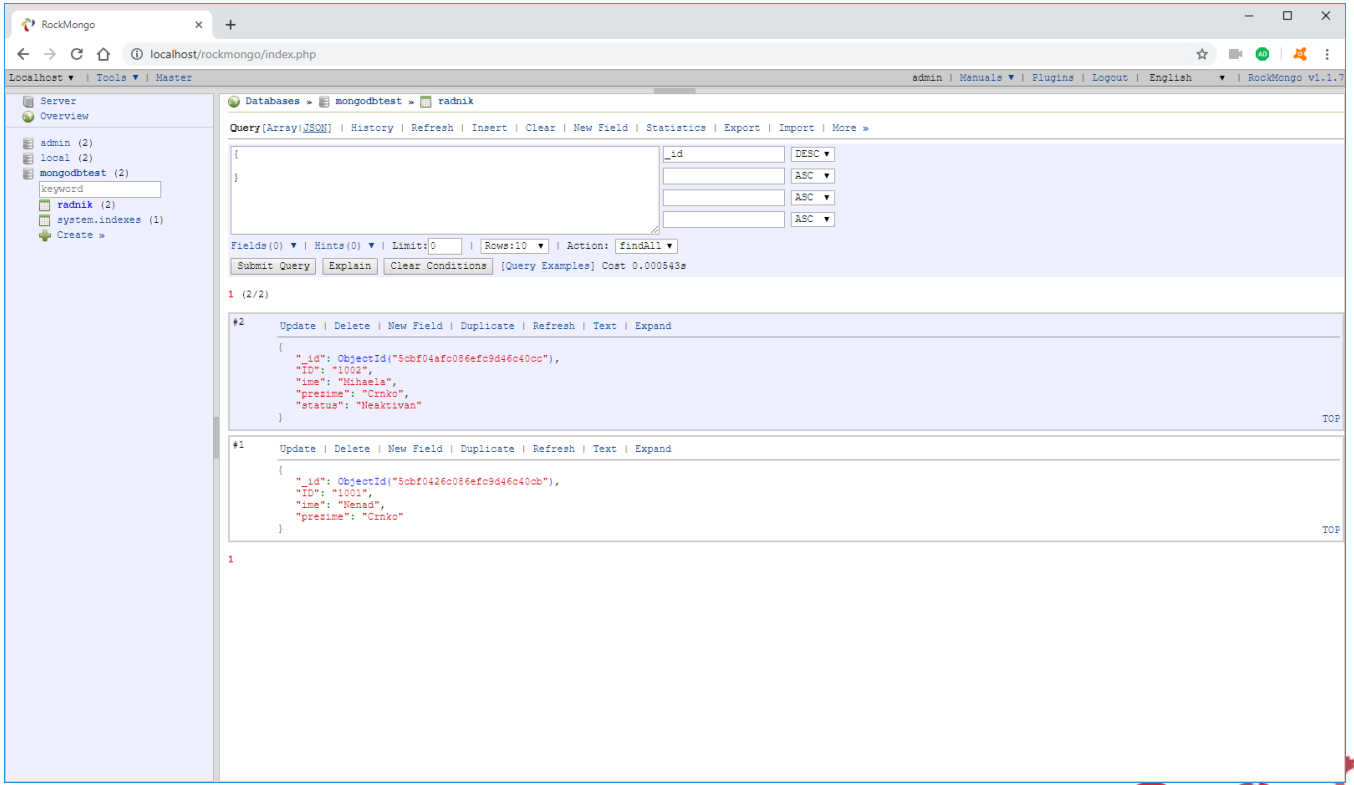
Primjer rada s podacima - MongoDB

```
use mongodbtest;

db.radnik.insertOne(
   { ID: "1001", ime: "Nenad", prezime: "Crnko" }
);
db.radnik.insertOne(
   { ID: "1002", ime: "Mihaela", prezime: "Crnko", status:
"Neaktivan" }
);
```

Rezultati izvođenja – preko mongo.exe



Rezultati izvođenja – RockMongo



Dodatne operacije MySQL – čitanje, ažuriranje, brisanje

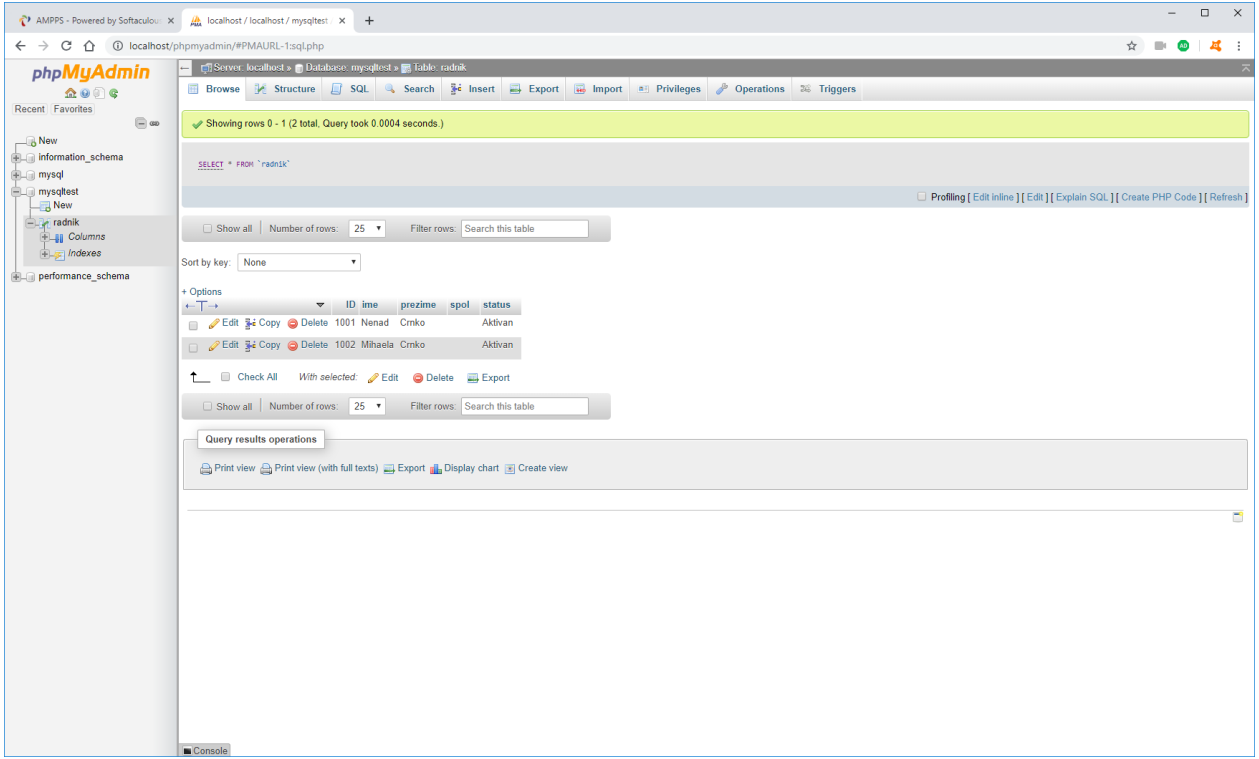
```
SELECT * from radnik where status = 'Aktivan';
```

```
UPDATE radnik set status = 'Aktivan' where status =
'Neaktivan';
```

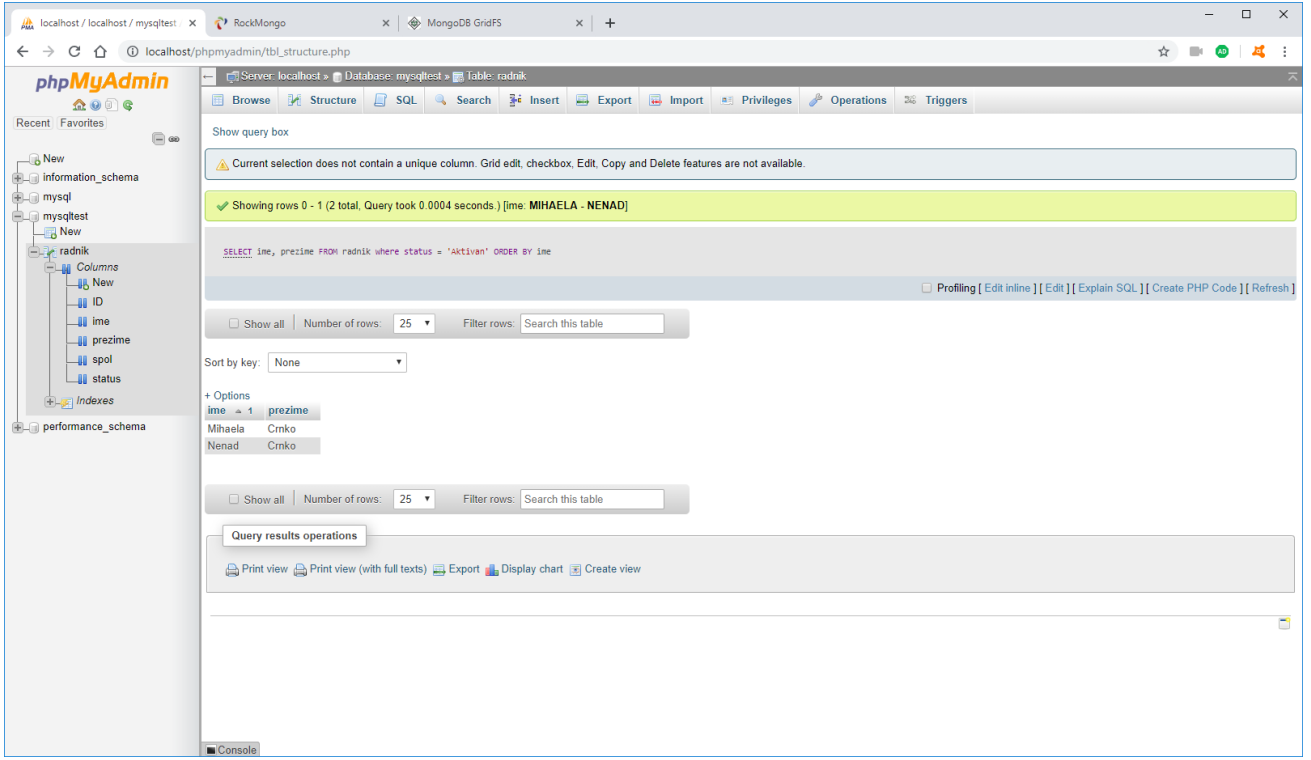
```
DELETE FROM radnik where status = 'Neaktivan';
```

```
SELECT ime, prezime FROM radnik where status = 'Aktivan'
order by ime;
```

Rezultati izvođenja



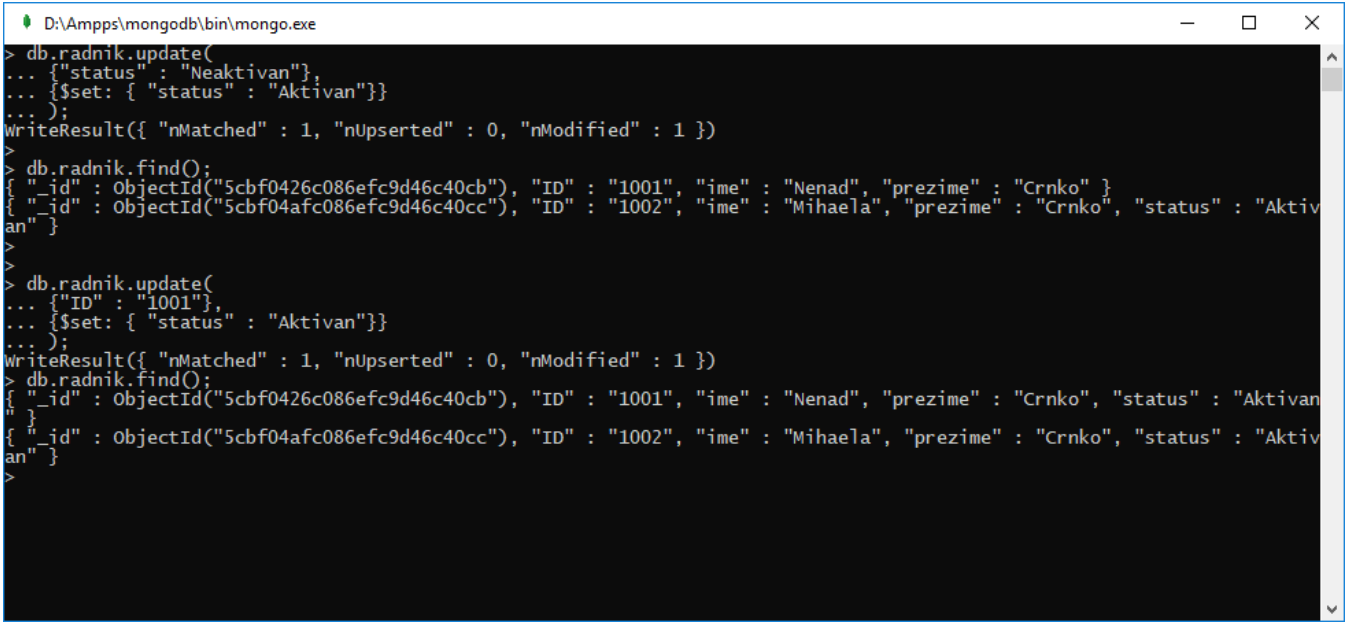
Rezultati izvođenja



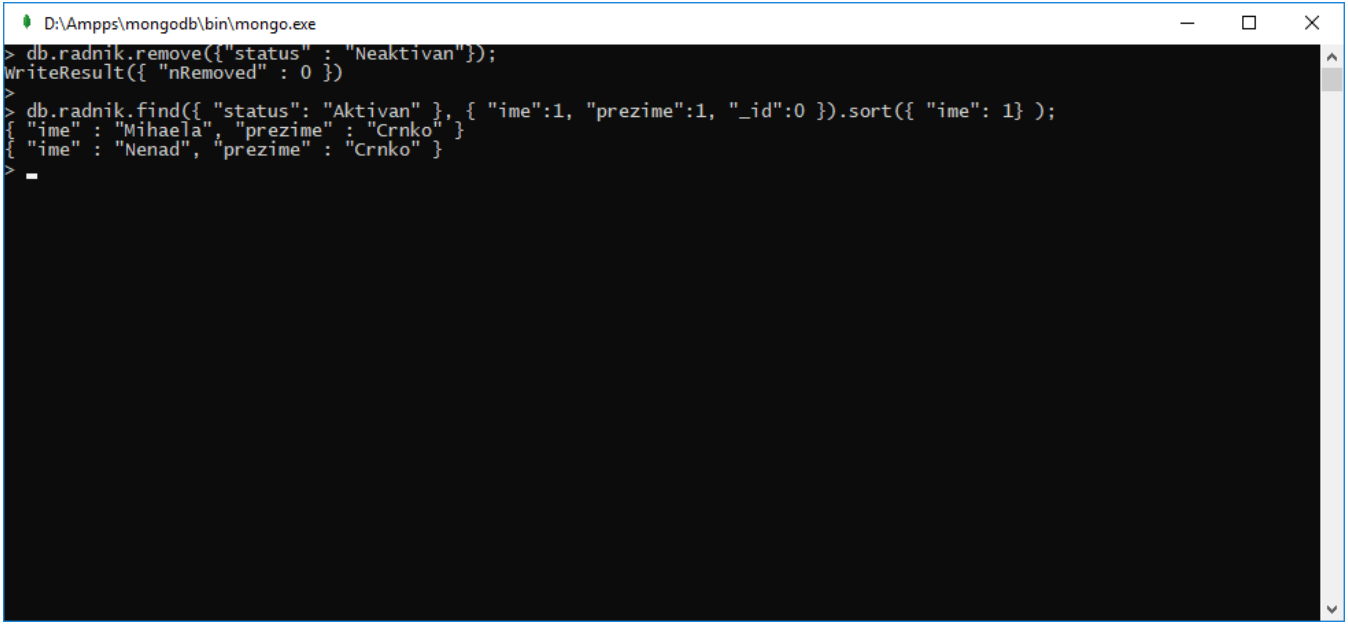
Dodatne operacije MongoDB – čitanje, ažuriranje, brisanje

```
db.radnik.update(
     {"status" : "Neaktivan"},
     {$set: { "status" : "Aktivan"}}
);

db.radnik.find({ "status": "Aktivan" });

db.radnik.update(
     {"ID" : "1001"},
     {$set: { "status" : "Aktivan"}}
);


db.radnik.remove({"status" : "Neaktivan"});

db.radnik.find({ "status": "Aktivan" }, { "ime":1, "prezime":1,
"_id":0 }).sort({ "ime": 1} );
```

Rezultati izvođenja



Rezultati izvođenja



Rad s nestrukturiranim podacima

Dodatni primjeri MongoDB

Primjer upisa nekoliko dodatnih podataka s različitim atributima.

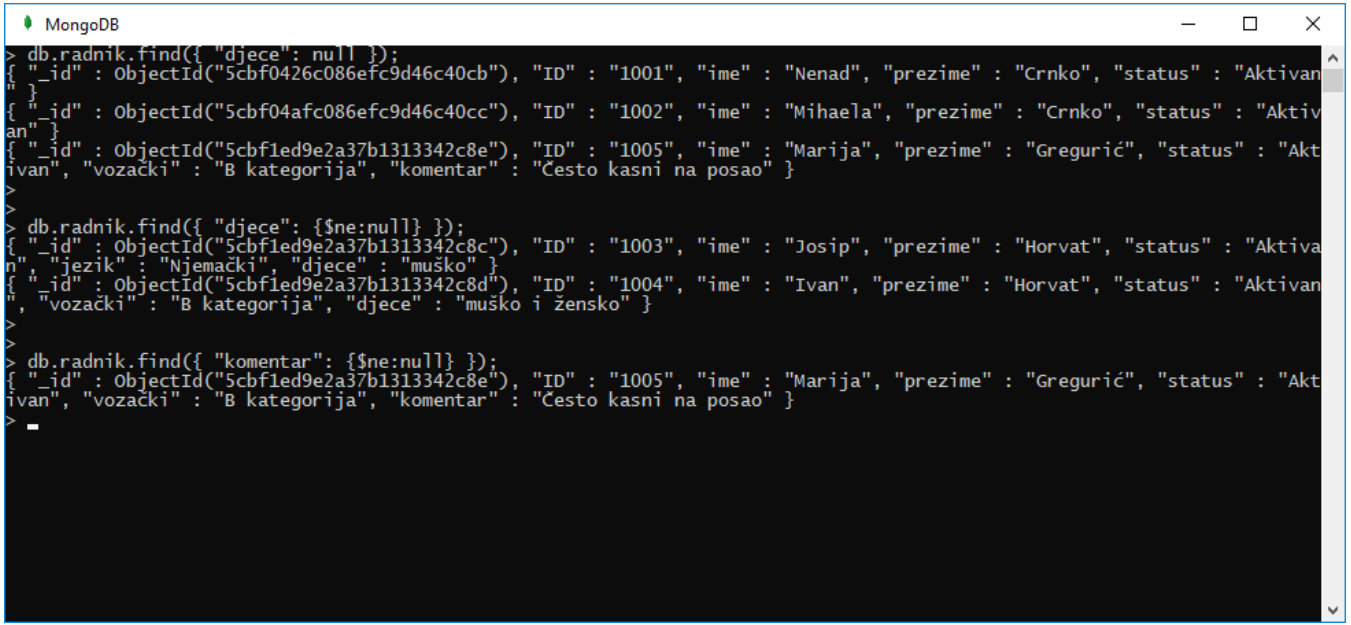
```
db.radnik.insertOne(
   { ID: "1003", ime: "Josip", prezime: "Horvat", status: "Aktivan", jezik: "Njemački", djece: "muško" }
);

db.radnik.insertOne(
   { ID: "1004", ime: "Ivan", prezime: "Horvat", status: "Aktivan", vozački: "B kategorija", djece: "muško i žensko" }
);

db.radnik.insertOne(
   { ID: "1005", ime: "Marija", prezime: "Gregurić", status: "Aktivan", vozački: "B kategorija", komentar: "Često kasni na posao" }
);
```

Dodatni primjeri MongoDB

Korištenje *null* vrijednosti kod pretraživanja.

```
db.radnik.find({ "djece": null });

db.radnik.find({ "djece": {$ne:null} });

db.radnik.find({ "komentar": {$ne:null} });
```

Rezultati izvođenja



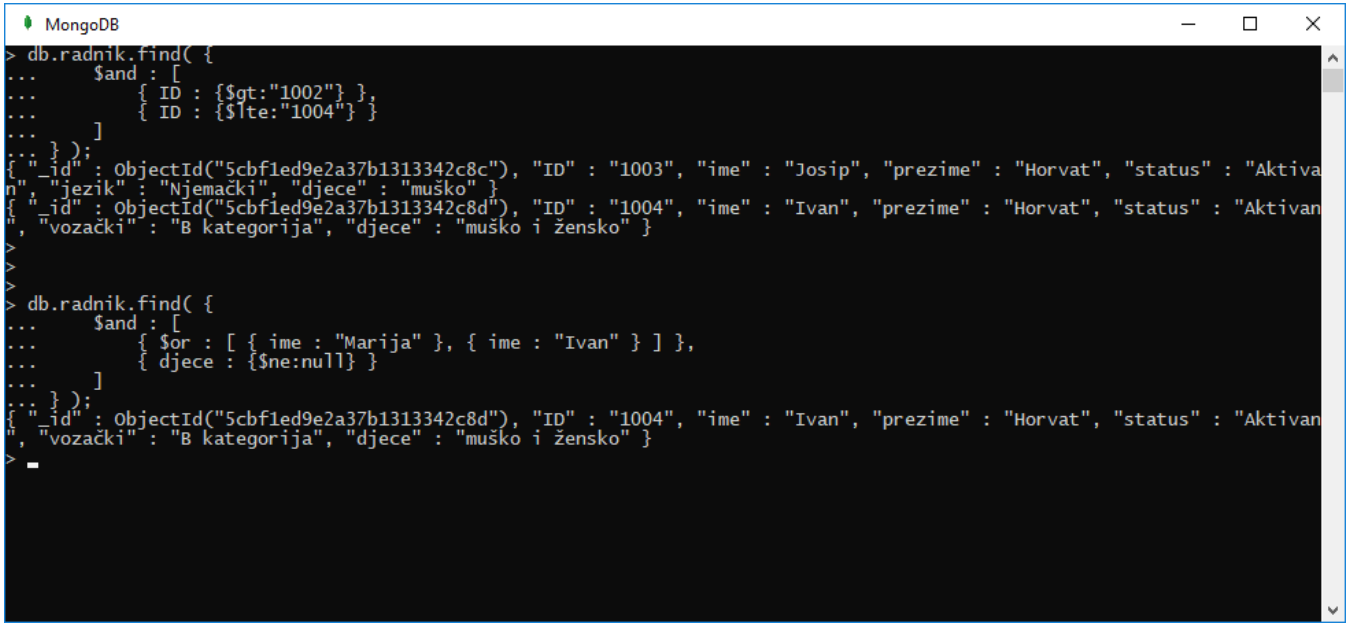
Dodatni primjeri MongoDB

Korištenje operatora kod pretraživanja.

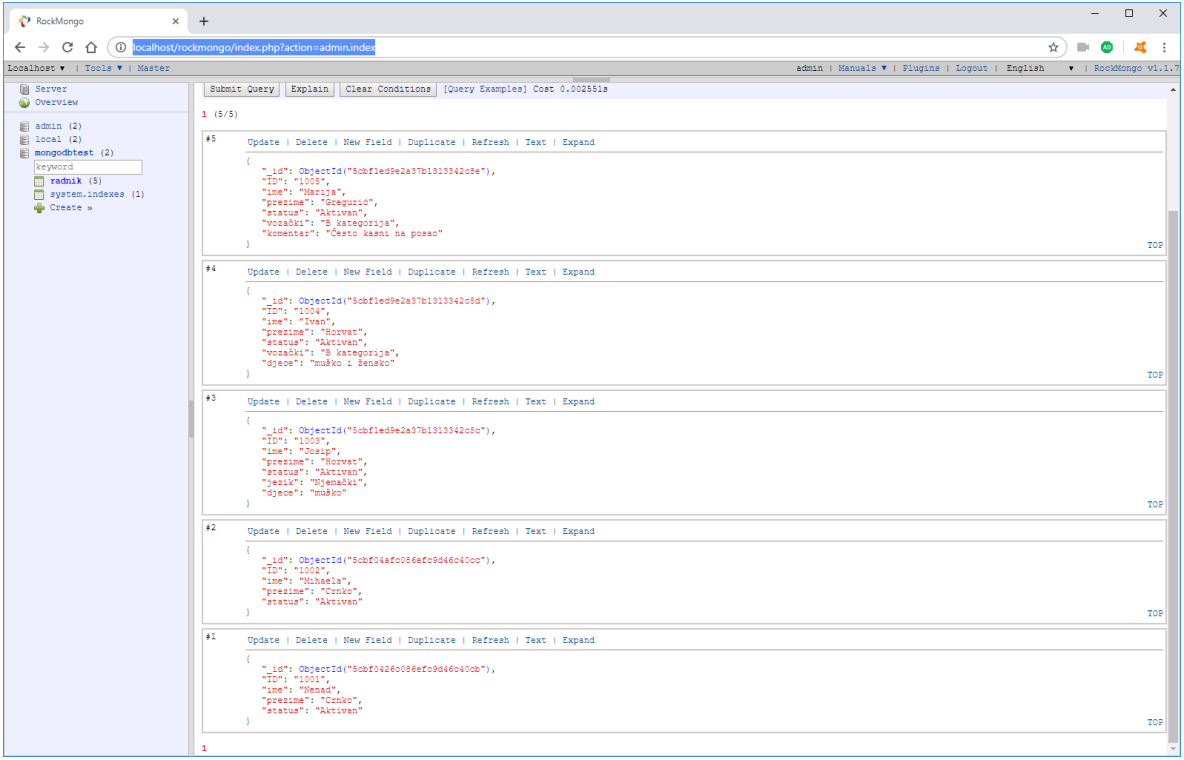
```
db.radnik.find( {
    $and : [
        { $or : [ { ime : "Marija" }, { ime : "Ivan" } ] },
        { djece : {$ne:null} }
    ]
} );
```

```
db.radnik.find( {
    $and : [
        { ID : {$gt:"1002"} },
        { ID : {$lte:"1004"} }
    ]
} );
```

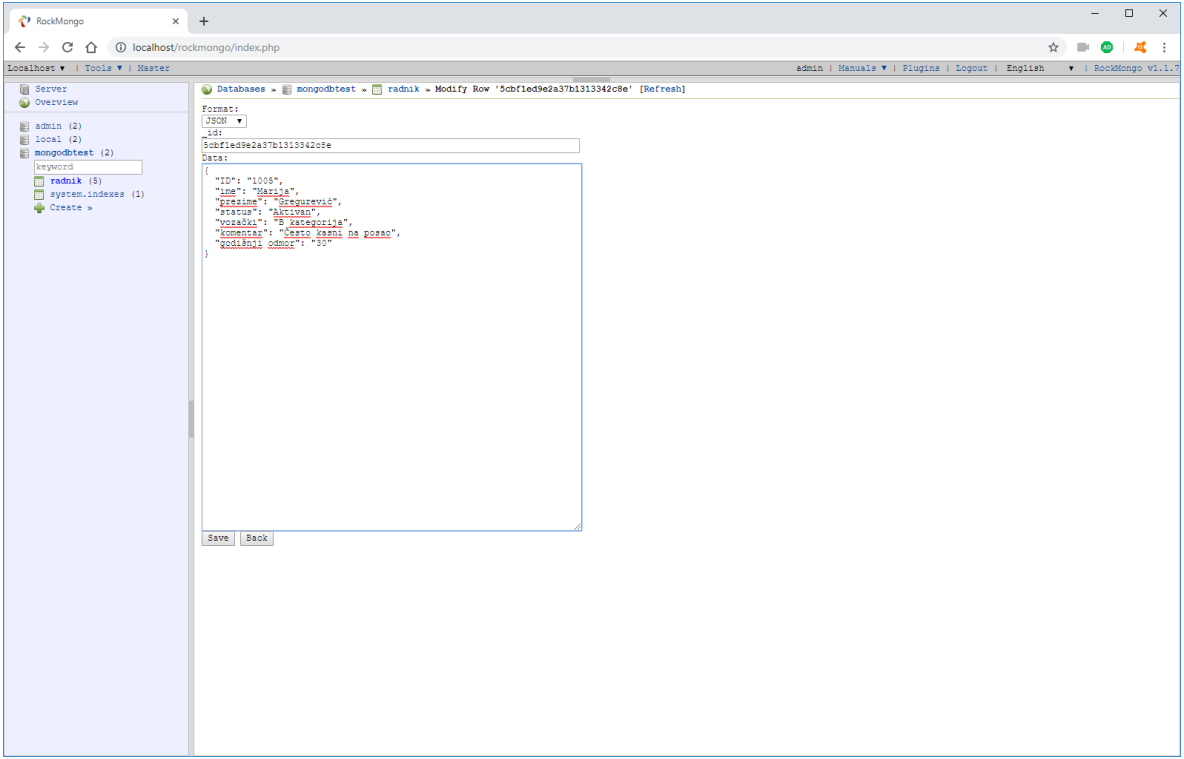
Rezultati izvođenja



Pregled podataka – RockMongo



Ažuriranje podataka – izravno RockMongo



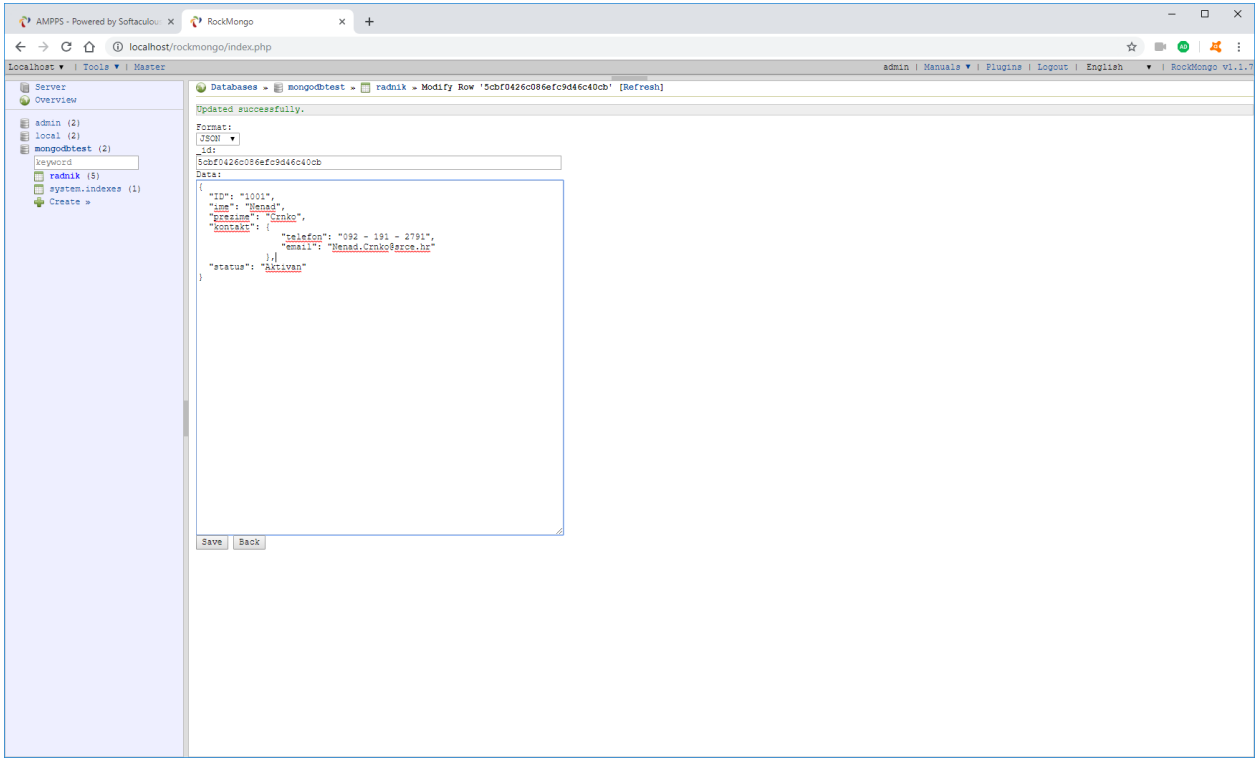
Složeniji oblici podataka

UMETNUTI (EMBEDDED) PODACI

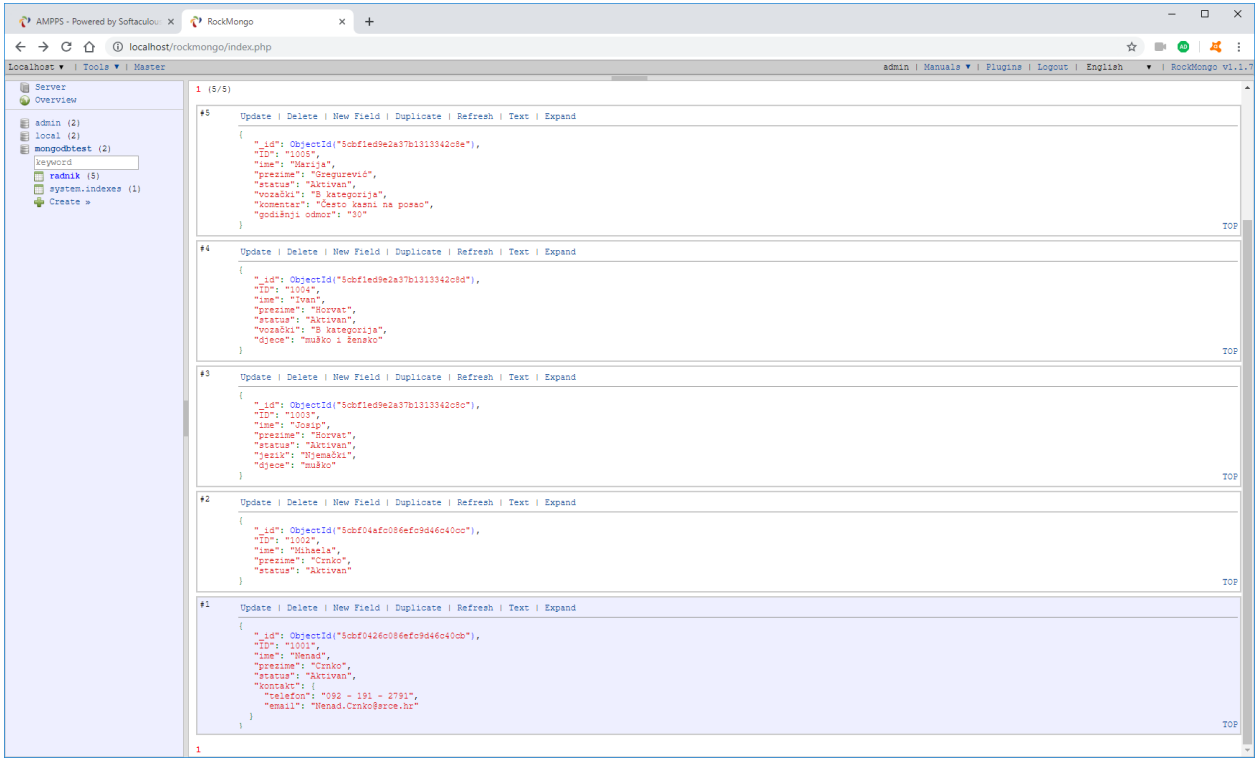
U osnovni dokument umetnuti su povezani podaci.

```
{
  "ID": "1001",
  "ime": "Nenad",
  "prezime": "Crnko",
  "kontakt": {
        "telefon": "092 - 191 - 2791",
        "email": "Nenad.Crnko@srce.hr"
         },
  "status": "Aktivan"
}
```

Pregled podataka – RockMongo



Pregled podataka – RockMongo



UMETNUTI (EMBEDDED) PODACI

Optimalan model organizacije podataka u većini slučajeva (**denormalizirani model podataka**). Dokument mora biti u okviru ograničenja za BSON dokument (16 MB i 100 razina). Za spremanje slika i drugih većih dokumenata koristi se poseban GridFS format.

Bolje performanse čitanja („sve je na jednom mjestu”), te garantirani zapis dokumenta kod ažuriranja.

Za pristup dokumentima s umetnutim podacima koristi se posebna sintaksa upita s dodatnom točkom.

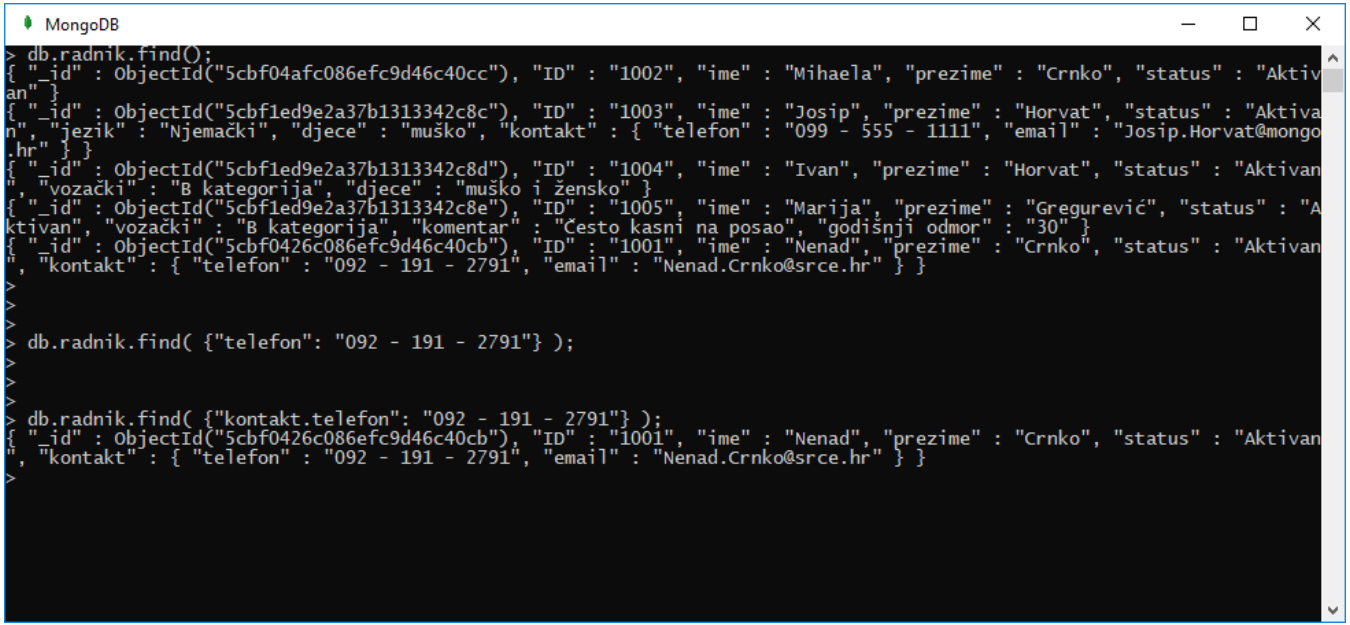
Dozvoljeno je korištenje dodatnih operatora kao kod osnovnih dokumenata.

PRISTUP UMETNUTIM PODACIMA

```
{
  "ID": "1003",
  "ime": "Josip",
  "prezime": "Horvat",
  "status": "Aktivan",
  "kontakt": {
         "telefon": "099 - 555 - 1111",
         "email": "Josip.Horvat@mongo.hr"
  },
  "jezik": "Njemački",
  "djece": "muško"
}
```

```
db.radnik.find();
db.radnik.find( {"telefon": "092 - 191 - 2791"} );  //!!!
db.radnik.find( {"kontakt.telefon": "092 - 191 - 2791"} );
```

Rezultati izvođenja



GRIDFS

GridFS definira način spremanja BSON dokumenata čija veličina prelazi 16 MB.

Veliki dokument se automatski dijeli na veći broj manjih dokumenata s veličinom od 255 KB. Kod pristupa dokumentima dijelovi se automatski spajaju u polazni dokument, ali se može pristupiti i određenom dijelu dokumenta bez potrebe za učitavanjem cijelog dokumenta u memoriju.

Za spremanje se automatski koriste dvije kolekcije:

files	Metapodaci o datotekama
chunks	Binarne vrijednosti za dijelove dokumenata.

GRIDFS – format zapisa

files

```
{
  "_id" : <ObjectId>,
  "length" : <num>,
  "chunkSize" : <num>,
  "uploadDate" : <timestamp>,
  "md5" : <hash>,
  "filename" : <string>,
  "contentType" : <string>,
  "aliases" : <string array>,
  "metadata" : <any>,
}
```

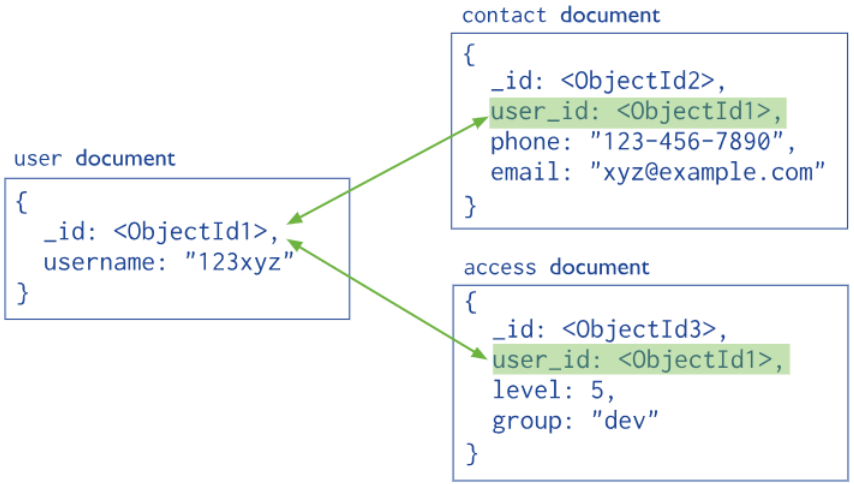
GRIDFS – format zapisa

chunks

```
{
  "_id" : <ObjectId>,
  "files_id" : <ObjectId>,
  "n" : <num>,
  "data" : <binary>
}
```

NORMALIZIRANI PODACI

U ovom slučaju se **_id** vrijednost jednog dokumenta koristi kao veza u drugom dokumentu.



NORMALIZIRANI PODACI

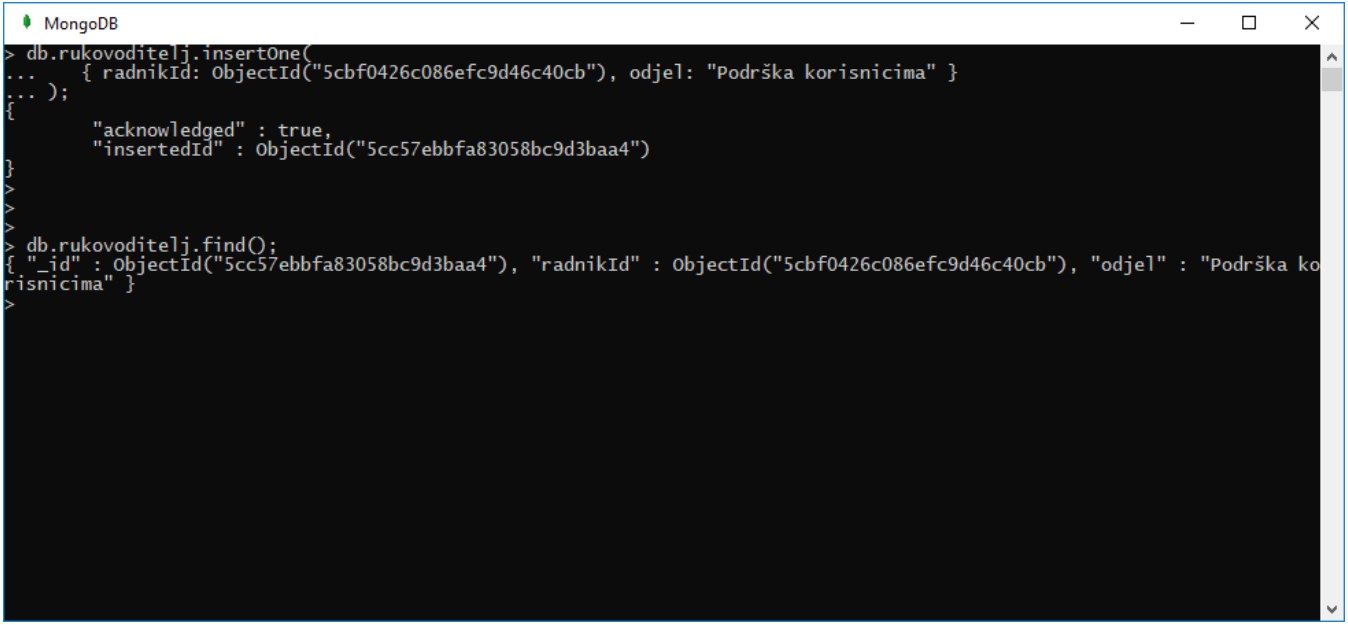
Koristi se u manjem broju vrlo složenih slučajeva podataka kad bi umetanje podataka izazvalo veliku količinu dupliciranih podataka bez povećanja brzine u izvođenju operacija s bazom podataka.

Korisnikova aplikacija treba slijediti veze da bi se organizirao pristup svim potrebnim podacima.

Primjer normalizacije dokumenata kod upotrebe dodatne kolekcije dokumenata.

```
db.rukovoditelj.insertOne(
   { radnikId: ObjectId("5cbf0426c086efc9d46c40cb"), odjel:
"Podrška korisnicima" }
);
db.rukovoditelj.find();
```

Rezultati izvođenja

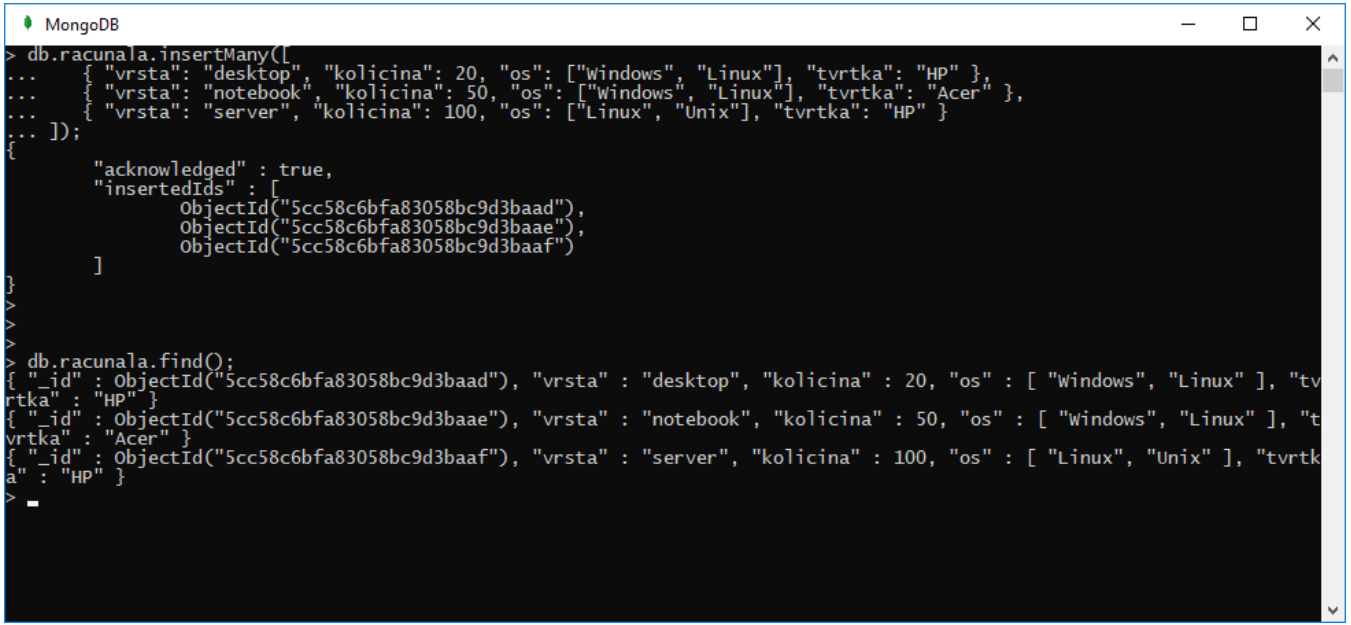


KORIŠTENJE POLJA VRIJEDNOSTI

U kolekciji dokumenata mogu se koristiti i polja vrijednosti. Na primjer:

```
db.racunala.insertMany([
   { "vrsta": "desktop", "kolicina": 20, "os": ["Windows",
"Linux"], "tvrtka": "HP" },
   { "vrsta": "notebook", "kolicina": 50, "os": ["Windows",
"Linux"], "tvrtka": "Acer" },
   { "vrsta": "server", "kolicina": 100, "os": ["Linux",
"Unix"], "tvrtka": "HP" }
]);

db.racunala.find();
```

Rezultati izvođenja

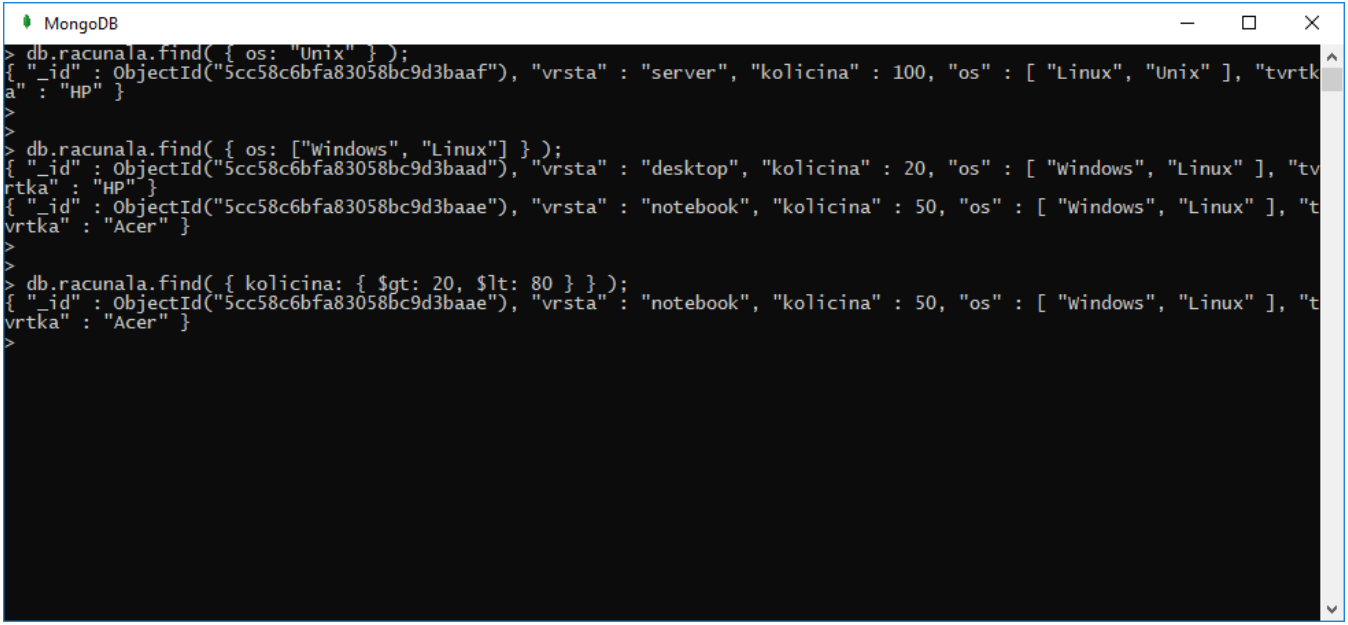


KORIŠTENJE POLJA

Nekoliko primjera postavlja upita prema poljima:

```
db.racunala.find( { os: "Unix" } );

db.racunala.find( { os: ["Windows", "Linux"] } );

db.racunala.find( { kolicina: { $gt: 20, $lt: 80 } } );
```

Rezultati izvođenja



IZVOĐENJE NAREDBI

MongoDB **garantira** izvođenje operacije zapisivanja pojedinačnog dokumenta čak i u slučajevima kad operacija mijenja više umetnutih dokumenata unutar osnovnog dokumenta.

Kod zapisivanja podataka u veći broj dokumenata **ne garantira** se zapis kod izvođenja različitih „Many” operacija.

Za garantirano ispravno izvođenje „Many” operacija potrebno je koristiti **transakcije**, koje su dostupne od verzije 4.0.

Transakcije koje se izvode na većem broju dokumenata ponašaju se jednako kao i transakcije koje se izvode na većem broju tablica u MySQL.

Transakcije se mogu izvoditi samo na postojećim kolekcijama (koje mogu biti u različitim bazama), te nije dozvoljeno korištenje na sistemskim bazama.

PRIMJER KORIŠTENJA TRANSAKCIJE

```
session = db.getMongo().startSession( { readPreference: {
mode: "primary" } } );


employeesCollection = session.getDatabase("hr").employees;
eventsCollection = session.getDatabase("reporting").events;
session.startTransaction( { readConcern: { level: "snapshot"
}, writeConcern: { w: "majority" } } );
try {
   employeesCollection.updateOne( { employee: 3 }, { $set: {
status: "Inactive" } } );
   eventsCollection.insertOne( { employee: 3, status: { new:
"Inactive", old: "Active" } } );
} catch (error) {
   session.abortTransaction();
   throw error;
}
session.commitTransaction();
session.endSession();
```

PREDUVJETI ZA TRANSAKCIJE

MongoDB verzija 4.0 ili novija (trenutna verzija 4.0.9).

Odgovarajuća verzija upravljačkih programa za različite programske jezike.

Java 3.8.0	C# 2.7	Perl 2.0.0
Python 3.7.0	Node 3.1.0	PHPC 1.5.0
C 1.11.0	Ruby 2.6.0	Scala 2.4.0

Optimizacija korištenja

MySQL

https://sistemac.srce.hr/seminar-za-it-specijaliste-optimiziranje-mysql-baze-i-laravela-za-rad-s-vrlo-velikim-bazama-podataka

Optimizacija sustava za upravljanje bazom podataka (MySQL)
Storage engine (vrste i odabir), međusobno pretvaranje tablica i ostalo

Optimizacija logičkog modela baze podataka
Optimalno korištenje različitih vrsta podataka, normalizirane i denormalizirane tablice, optimalno korištenje indeksa, defragmentacija tablica i indeksa

Optimizacija upita na bazu podataka
Pravila za pisanje upita, parametri za optimizaciju konfiguracije sustava za upravljanje bazom podataka.

MongoDB – STORAGE ENGINE

WIREDTIGER STORAGE ENGINE (podrazumijevana)

Podrazumijevani sistemski modul od verzije 3.2 pa samim tim i preporučen za korištenje. Nudi zaključavanje kod zapisa na razini dokumenta, kontrolne točke (svakih 60 sec), sažimanje, itd.

IN-MEMORY STORAGE ENGINE

Dostupno samo u Enterprise izdanju. Umjesto spremanja dokumenata na disku intenzivno se koristi memorije što povećava brzinu rada.

MMAPv1 Storage Engine (podrazumijevano do verzije 30.)

Više se ne preporuča korištenje.

PROVJERA ISPRAVNOSTI DOKUMENATA

Dostupna je od verzije 3.6 i koristi se za definiranje zahtijevanog oblika dokumenata (JSON Schema validation).

Primjer:

```
db.createCollection("studenti", {
   validator: {
      $jsonSchema: {
         bsonType: "object",
         required: [ "ime", "prezime", "godina", "adresa.grad",
"adresa.ulica" ],
         properties: {
            ime: {
               bsonType: "string",
               description: "Obavezan unos imena u obliku niza znakova"
            },
            prezime: {
               bsonType: "string",
               description: "Obavezan unos prezimena u obliku niza znakova"
            },
```

PROVJERA ISPRAVNOSTI DOKUMENATA

```
 godina: {
                bsonType: "int",
                minimum: 2019,
                maximum: 2025,
                exclusiveMaximum: false,
                description: "Obavezan unos cijelog broja u rasponu [ 2017,
3017 ]"
             },
             "adresa.grad" : {
                bsonType: "string",
                description: "Obavezan unos grada u obliku niza znakova"
             },
             "adresa.ulica" : {
                bsonType: "string",
                description: "Obavezan unos ulice u obliku niza znakova"
             }
          }
       }
    }
})
```

PROVJERA ISPRAVNOSTI DOKUMENATA

Razine provjere (validationLevel):

strict primijenjuje se kod svih operacija zapisa dokumenata

moderate samo na već spremljene dokumente koji zadovoljavaju provjeru

Akcije nakon provjere (validationAction):

error odbacuju se dokumenti koji ne zadovoljavaju pravila

warn dozvoljava se zapis uz bilježenje upozorenja

MongoDB - INDEKSI

Bez korištenja indeksa izvodi se detaljno pregledavanje dokumenata u odgovarajućoj kolekciji (scan kolekcije). Nešto slično tablicama kod relacijskih orijentiranih baza podataka.

Korištenjem indeksa smanjuje se broj dokumenata koje treba pregledati tijekom izvođenja upita.

Na svakoj kolekciji se automatski priprema jedinstveni indeks na polju _**id**, čime se sprečava stvaranje dva dokumenta s istom vrijednošću ovog stupca.

JEDNOSTAVNI INDEKSI

Pripremaju se na jednom podatku u okviru dokumenta. Moguće ih je pripremiti u rastućem ili padajućem redoslijedu (1, -1).

```
{
  "_id": ObjectId("570c04a4ad233577f97dc459"),
  "score": 1034,
  "location": { state: "NY", city: "New York" }
}
db.records.createIndex( { score: 1 } )

    db.records.find( { score: 2 } )
    db.records.find( { score: { $gt: 10 } } )

db.records.createIndex( { "location.state": 1 } )

    db.records.find( { "location.state": "CA" } )
    db.records.find( { "location.city": "Albany",
"location.state": "NY" } )
```

SLOŽENI INDEKSI

Pripremaju se na većem broju podataka u okviru dokumenta.

```
{
 "_id": ObjectId(570c04a4ad233577f97dc432),
 "item": "Banana",
 "category": ["food", "produce", "grocery"],
 "location": "4th Street Store",
 "stock": 4,
 "type": "cases"
}
db.products.createIndex( { "item": 1, "stock": 1 } )
    db.products.find( { item: "Banana" } )
    db.products.find( { item: "Banana", stock: { $gt: 5 }
} )
```

Mogućnost korištenja kod pretraživanja je slična kao kod relacijskih indeksa (s lijeva na desno).

SLOŽENI INDEKSI

U slučaju da se indeks želi pripremiti po podatku u dokumentu koji u predstavlja polje vrijednosti , automatski se priprema indeks tako da obuhvaća svaki element u polju.

```
{ _id: 1, item: "ABC", ratings: [ 2, 5, 9 ] }
```

```
db.survey.createIndex( { ratings: 1 } )
```

TEKST INDEKSI

Koriste se za potrebe pretraživanja velikih nizova znakova (slično kao kod relacijskih baza podataka).

Tijekom razvoja MongoDB baze pojavilo se nekoliko verzija (1, 2, 3) čije korištenje se može navesti kod pripreme indeksa.

```
db.reviews.createIndex(
    {
      subject: "text",
      comments: "text"
    }
 )
```

TEKST INDEKSI

Kod pripreme indeksa može se dodatno ograničiti pretraživanje prema drugom podatku.

```
{ _id: 1, dept: "tech", description: "lime green computer" }
{ _id: 2, dept: "tech", description: "wireless red mouse" }
{ _id: 3, dept: "kitchen", description: "green placemat" }
{ _id: 4, dept: "kitchen", description: "red peeler" }
{ _id: 5, dept: "food", description: "green apple" }
{ _id: 6, dept: "food", description: "red potato" }

db.inventory.createIndex(
   {
     dept: 1,
     description: "text"
   }
)
```

```
db.inventory.find( { dept: "kitchen", $text: { $search: "green" } } )
```

HASH INDEKSI

Kod pripreme indeksa koristi se hash funkcija za izračunavanje vrijednosti indeksiranog podatka.

Koriste se i kod podjele podataka na veći broj servera klastera.

Kod pretraživanja korištenjem hash indeksa aplikacija ne treba nezavisno pripremati istu vrijednost.

Tijekom operacije pripreme hash vrijednosti brojevi s pokretnim zarezom se svode na 64 bitne cjelobrojne brojeve pa različite vrijednosti mogu dobiti istu hash vrijednost.

```
db.collection.createIndex( { _id: "hashed" } )
```

2DSPHERE INDEKSI

Omogućavaju pripremu posebne vrste indeksa koji se koriste kod “geoorijentiranih” podataka. Na primjer: MultiPoint, MultiLineString, MultiPolygon, i GeometryCollection

```
db.places.insert(
   {
      loc : { type: "Point", coordinates: [ -73.97, 40.77 ] },
      name: "Central Park",
      category : "Parks"
   }
)
db.places.insert(
   {
      loc : { type: "Point", coordinates: [ -73.88, 40.78 ] },
      name: "La Guardia Airport",
      category : "Airport"
   }
)
    db.places.createIndex( { loc : "2dsphere" } )
```

DODATNA SVOJSTVA INDEKSA

TTL indeksi

Posebna vrsta jednostavnih indeksa koji se mogu koristiti za automatsko uklanjanje dokumenta iz kolekcije nakon određenog vremena.

```
db.eventlog.createIndex( { "lastModifiedDate": 1 }, { expireAfterSeconds: 3600 } )
```

Brisanje indeksa se izvodi u posebnom pozadinskom procesu koji se ponavlja svakih 60 sekundi, tako da pojedini dokumenti mogu ostati i malo duže u bazi od planiranog vremena.

DODATNA SVOJSTVA INDEKSA

Jedinstveni indeksi

Sprečavaju spremanje ponavljajućih vrijednosti određenog podatka.

Podrazumijevano takvo svojstvo ima indeks koji se priprema za _id stupac.

Može se koristiti kod pripreme jednostavnih i kod pripreme složenih indeksa.

```
db.members.createIndex( { "user_id": 1 }, { unique: true } )
```

```
db.members.createIndex( { groupNumber: 1, lastname: 1,
firstname: 1 }, { unique: true } )
```

DODATNA SVOJSTVA INDEKSA

Djelomični indeksi

Pripremaju samo za one dokumente iz kolekcije koji zadovoljavaju određeni kriterij.

Zahtijevaju manje resursa i prostora od standardnih indeksa, a pomažu kod određene vrste najčešćih upita.

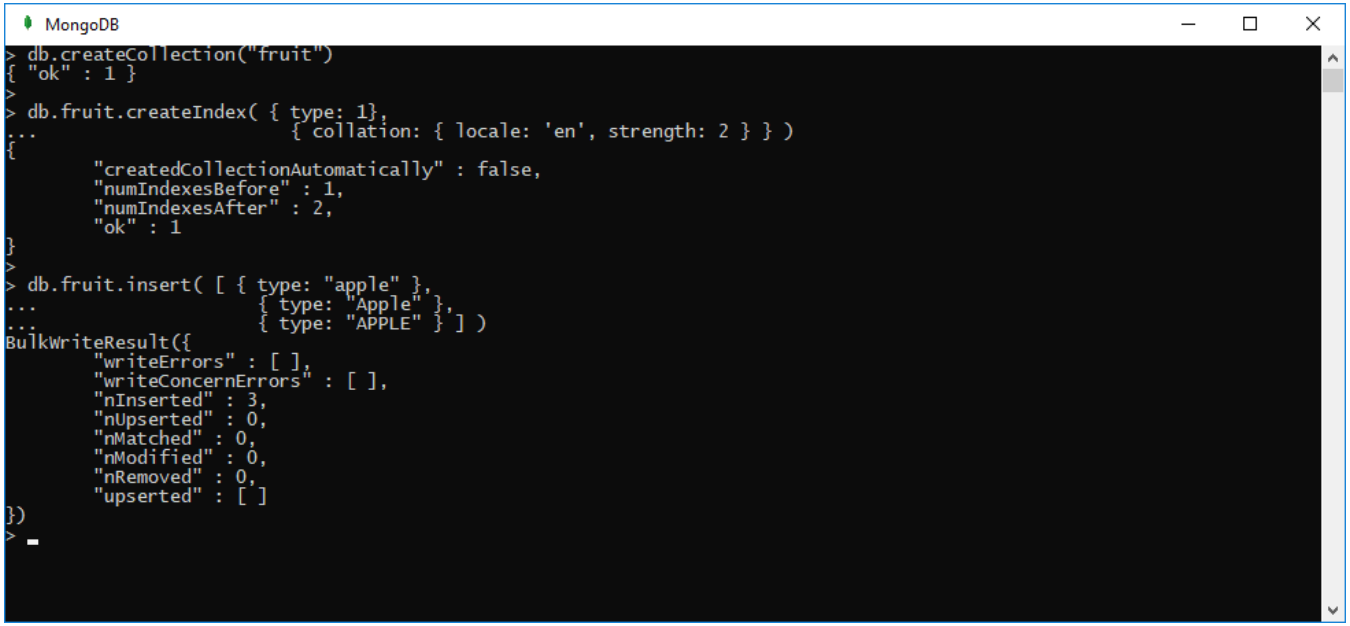
```
db.restaurants.createIndex(
   { cuisine: 1, name: 1 },
   { partialFilterExpression: { rating: { $gt: 5 } } }
)
```

DODATNA SVOJSTVA INDEKSA

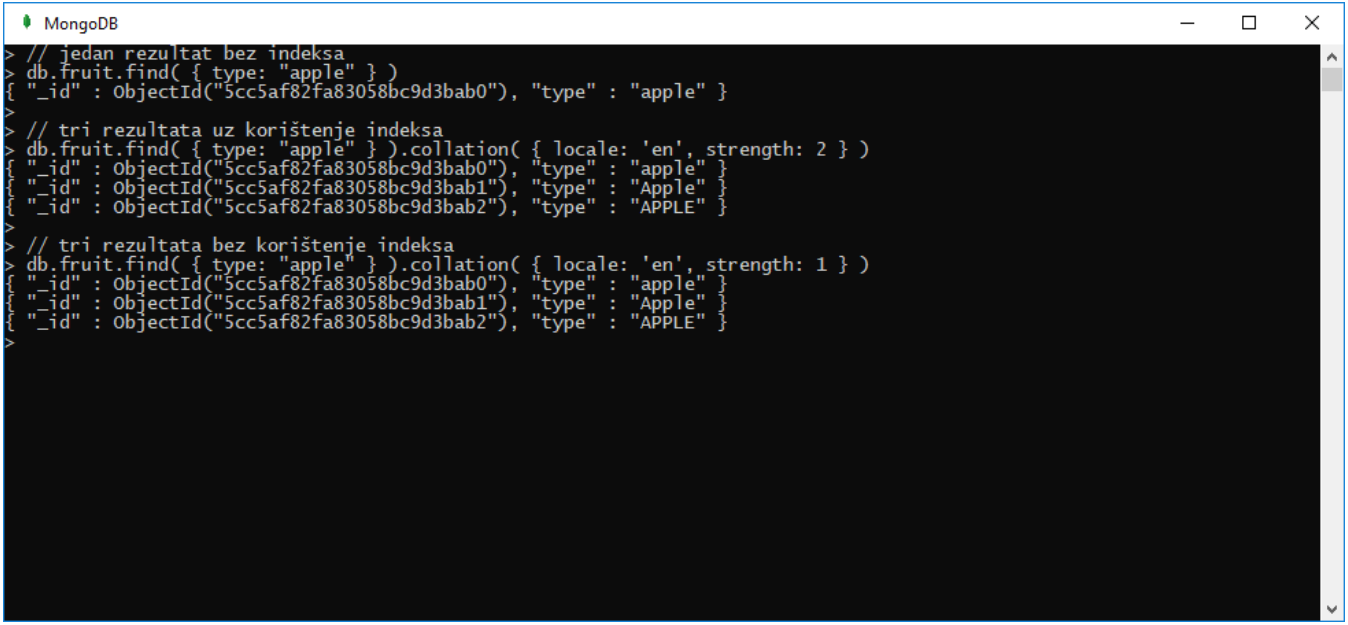
Prilikom kreiranja mogu se definirati dodatna svojstva koja se mogu koristiti i kod pretraživanja dokumenata uz navođenje tih istih dodatnih karakteristika indeksa.

```
// jedan rezultat bez indeksa
db.fruit.find( { type: "apple" } )

// tri rezultata uz korištenje indeksa
db.fruit.find( { type: "apple" } ).collation( { locale: 'en',
strength: 2 } )

db.fruit.find( { type: "apple" } ).collation( { locale: 'en',
strength: 1 } )
// tri rezultata bez korištenje indeksa
```

Rezultati izvođenja



Rezultati izvođenja



ANALIZA KORIŠTENJA INDEKSA

```
db.fruit.find( { type: "apple" } ).collation( { locale: 'en', strength: 1 } ).explain();
```

```
{
        "queryPlanner" : {
                "plannerVersion" : 1,
                "namespace" : "mongodbtest.fruit",
                "indexFilterSet" : false,
                "parsedQuery" : {
                        "type" : {
                                "$eq" : "apple"
                        }
                },
                "collation" : {
                        "locale" : "en",
                        "caseLevel" : false,
                        "caseFirst" : "off",
                        "strength" : 1,
                        "numericOrdering" : false,
                        "alternate" : "non-ignorable",
                        "maxVariable" : "punct",
                        "normalization" : false,
                        "backwards" : false,
                        "version" : "57.1"
                },
```

ANALIZA KORIŠTENJA INDEKSA

```
"winningPlan" : {
                        "stage" : "COLLSCAN",
                        "filter" : {
                                "type" : {
                                        "$eq" : "apple"
                                }
                        },
                        "direction" : "forward"
                },
                "rejectedPlans" : [ ]
        },
        "serverInfo" : {
                "host" : "HP640G4",
                "port" : 27017,
                "version" : "3.4.7",
                "gitVersion" : "cf38c1b8a0a8dca4a11737581beafef4fe120bcd"
        },
        "ok" : 1
}
```

ANALIZA KORIŠTENJA INDEKSA

```
db.fruit.find( { type: "apple" } ).collation( { locale: 'en', strength: 2 } ).explain();
```

```
{
        "queryPlanner" : {
                "plannerVersion" : 1,
                "namespace" : "mongodbtest.fruit",
                "indexFilterSet" : false,
                "parsedQuery" : {
                        "type" : {
                                "$eq" : "apple"
                        }
                },
                "collation" : {
                        "locale" : "en",
                        "caseLevel" : false,
                        "caseFirst" : "off",
                        "strength" : 2,
                        "numericOrdering" : false,
                        "alternate" : "non-ignorable",
                        "maxVariable" : "punct",
                        "normalization" : false,
                        "backwards" : false,
                        "version" : "57.1"
                },
```

ANALIZA KORIŠTENJA INDEKSA


```
"winningPlan" : {
                        "stage" : "FETCH",
                        "inputStage" : {
                                "stage" : "IXSCAN",
                                "keyPattern" : {
                                        "type" : 1
                                },
                                "indexName" : "type_1",
                                "collation" : {
                                        "locale" : "en",
                                        "caseLevel" : false,
                                        "caseFirst" : "off",
                                        "strength" : 2,
                                        "numericOrdering" : false,
                                        "alternate" : "non-ignorable",
                                        "maxVariable" : "punct",
                                        "normalization" : false,
                                        "backwards" : false,
                                        "version" : "57.1"
                                },
                                "isMultiKey" : false,
                                "isUnique" : false,
                                "isSparse" : false,
                                "isPartial" : false,
                                "indexVersion" : 2,
                                "direction" : "forward",
```

ANALIZA KORIŠTENJA INDEKSA


```
 "indexBounds" : {
                                        "type" : [
"[\")GG?1\u0001\t\", \")GG?1\u0001\t\"]"
                                                  ]
                                    }
                            }
                },
 "rejectedPlans" : [ ]
        },
        "serverInfo" : {
                "host" : "HP640G4",
                "port" : 27017,
                "version" : "3.4.7",
                "gitVersion" :
"cf38c1b8a0a8dca4a11737581beafef4fe120bcd"
        },
        "ok" : 1
```

PREGLED I UKLANJANJE INDEKSA

```
db.radnik.getIndexes();

[
        {
                "v" : 2,
                "key" : {
                        "_id" : 1
                },
                "name" : "_id_",
                "ns" : "mongodbtest.radnik"
        }
]

db.radnik.dropIndexes()

db.radnik.dropIndexes(„_id_")
```

„MySQL vraća udarac”

ZAŠTO I KADA I DALJE KORISTITI MYSQL

Iako u određenim situacijama MongoDB (i drugi alternativni modeli) imaju prednosti u odnosu na relacijski model, to ne znači da će MySQL i ostale relacijske baze podataka odjednom nestati iz upotrebe.

U relacijskim bazama već se čuva velika količina podataka, a brojna aplikativna rješenja pripremljena su isključivo za njihovu upotrebu. Prelazak na alternativne modele tražio bi ogromne resurse (tehničke, ljudske i financijske).

Često je potpuno zadovoljavajuće rješenje očuvanje relacijskog modela baze podataka uz nadogradnju fleksibilnijom mogućnošću čuvanja nestrukturiranih podataka.

MySQL za to nudi vrstu podataka JSON od verzije 5.7.8. Omogućava zapis i pretraživanje vrijednosti bez mogućnosti pripreme indeksa.

MYSQL JSON PRIMJER

```
CREATE TABLE weblog(
  id int auto_increment primary key,
  operation varchar(255),
  user varchar(255),
  page json,
  client json
);

INSERT INTO weblog(operation, user, page, client)
VALUES (
  'pageview',
   '1',
   '{ "page": "/" }',
   '{ "name": „Chrome", "os": „Windows", "resolution": { "x":
1600, "y": 900 } }'
),
```

MYSQL JSON PRIMJER

```
SELECT id, client->'$.name' client
FROM weblog;

SELECT id, client->>'$.name' client
FROM weblog;

SELECT client->>'$.name' client,
      count(browser)
FROM weblog
GROUP BY client->>'$.name';
```

KAKO IZABRATI BAZU PODATAKA

Ako se planiraju koristiti samo dobro strukturirani podaci, onda relacijski orijentirani sustav predstavlja bolje rješenje.

Ako je samo dio podataka nestrukturiran, a većina strukturirana, ili postoji „legacy” sustav koji treba nadograditi podrškom za nestrukturirane podatke, onda je najbolje izabrati relacijski model s proširenjem za korištenje nestrukturiranih podataka.

Za razvoj novog sustava s pretežno nestrukturiranim podacima je bolje izabrati MongoDB ili neki drugi NoSQL model baze podataka.

Suvremeni progbramski jezici omogućavaju različite “međuvarijante”.

Primjeri korištenja u različitim programskim jezicima

PRIMJERI KORIŠTENJA

Node.js

https://sistemac.srce.hr/seminar-za-it-specijaliste-uvod-u-graphql-s-primjerima-koristenja-u-razlicitim-tehnologijama-136

PHP

Priprema baze, kolekcije i zapis podataka u MongoDB

Python

Čitanje i prikaz podataka iz MongoDB, prepisivanje podataka između kolekcija, zapis slika.

Visual Studio

Pristup MongoDB i podacima preko vlastitog korisničkog sučelja.

Komentari i pitanja?

srce
Sveučilište u Zagrebu
Sveučilišni računski centar

Srce politikom otvorenog pristupa široj javnosti osigurava dostupnost i korištenje svih rezultata rada Srca, a prvenstveno obrazovnih i stručnih informacija i sadržaja nastalih djelovanjem i radom Srca.

www.srce.unizg.hr

creativecommons.org/licenses/by-nc/4.0/deed.hr

www.srce.unizg.hr/otvoreni-pristup

srce
otvoreni pristup

Sažetak predavanja

Osnovne karakteristike relacijskih (SQL) i nerelacijskih (NoSQL) baza podataka i prikaz njihovih razlika na primjerima baza MySQL i MongoDB

Izvođenje osnovnih operacija kao što su čitanje, upisivanje, ažuriranje i brisanje zapisa u bazi podataka

Naprednije operacije poput optimizacije tablica, indeksa, upita i ostalo

Primjeri korištenja uz pomoć nekoliko programskih jezika (PHP, Python i Visual Studio)