



Seminarski rad

Predmet: Sistemi za upravljanje bazama podataka

Tema: Interna struktura i organizacija MySQL skladišta podataka

Student: Jelena Đikić, br. ind. 1488

Niš, januar 2024

Sadržaj

1. Uvod.....	3
2. Sistemi za upravljanje bazama podataka	4
3. Interna struktura i organizacija sistema za upravljanje bazama podataka	5
4. MySQL sistem za upravljanje bazama podataka.....	6
5. Interna struktura i organizacija MySQL skladišta podataka	7
5.1. Aplikacioni sloj	8
5.1.1. Rukovanje vezom.....	9
5.1.2. Autentifikacija.....	12
5.1.3. Bezbednost	13
5.2. Logički sloj.....	15
5.2.1. Podsystem obrade upita	22
5.2.2. Upravljanje transakcijama.....	23
5.2.3. Menadžemnt oporavka	24
5.2.4. Menadžement skladištenja	25
5.3. Fizički sloj.....	26
5.3.1. Fajlovi sa podacima	27
5.3.2. Indeksi	28
5.3.3. Rečnik podataka.....	29
5.3.4. Statistički podaci	32
5.3.5. Log Informacije	34
6. Zaključak	35
7. Literatura	36

1. Uvod

Sistem za upravljanje bazama podataka (DBMS) zauzima ključno mesto u savremenim informacionim sistemima, pružajući temelj za efikasno skladištenje, organizaciju i upravljanje podacima. U dinamičkom okruženju današnjih tehnologija, ovaj seminarski rad ima za cilj da produbi razumevanje strukture i organizacije sistema za upravljanje bazama podataka, sa posebnim naglaskom na MySQL, jednom od popularnijih DBMS-ova u savremenom informacionom svetu.

Sistemi za upravljanje bazama podataka (engl. Database Management Systems) igraju ključnu ulogu u infrastrukturi za efikasno rukovođenje podacima u modernim aplikacijama. Prvi deo rada posvećen je osnovnim karakteristikama ovih sistema, duboko istražujući njihovu ulogu i značaj u kontekstu savremenih informacionih tehnologija. Razmatranje obuhvata ne samo njihovu ulogu u organizaciji podataka, već i doprinos u omogućavanju integriteta, sigurnosti i efikasnog pristupa podacima u različitim scenarijima.

Nakon uvodne analize SZUBP sistema uopšte, rad dalje istražuje MySQL sistem za upravljanje bazama podataka. Interna struktura i organizacija skladišta podataka su detaljno analizirane kroz slojeve: aplikacioni, logički i fizički. Fokus je posebno stavljen na logički sloj, ključnu komponentu koja se dalje dekomponuje na podsisteme obrade upita, upravljanja transakcijama, menadžmenta oporavka i menadžmenta skladištenja. Ovi podsistemi predstavljaju esencijalne gradivne blokove MySQL sistema, doprinoseći integrisanoj i efikasnoj manipulaciji podacima.

Razumevanje internih struktura SZUBP sistema, sa posebnim fokusom na MySQL, omogućava ne samo bolje iskorišćavanje njihovih funkcionalnosti već i efikasnije upravljanje podacima u širokom spektru aplikacija. Ovaj rad predstavlja korak dalje u otkrivanju složenosti i značaja sistema za upravljanje bazama podataka u današnjem digitalnom dobu.

2. Sistemi za upravljanje bazama podataka

Sistemi za upravljanje bazama podataka (engl. Database Management System - DBMS) su softverski sistemi koji se koriste za skladištenje, preuzimanje i pokretanje upita o podacima. DBMS služi kao interfejs između krajnjeg korisnika i baze podataka, omogućavajući korisnicima da kreiraju, čitaju, ažuriraju i brišu podatke u bazi podataka. DBMS upravlja podacima, mašinom baze podataka i šemom baze podataka, omogućavajući korisnicima i drugim programima da manipulišu podacima ili ih preuzimaju. Sistemi za upravljanje bazama podataka mogu se klasifikovati na osnovu različitih kriterijuma kao što su model podataka, distribucija baze podataka ili brojevi korisnika. Najrasprostranjeniji tipovi DBMS softvera su relacioni, distribuirani, hijerarhijski, objektno orijentisani i mrežni.

Distribuirani DBMS je skup logički međusobno povezanih baza podataka distribuiranih preko mreže kojom upravlja centralizovana aplikacija baze podataka. Ovaj tip DBMS-a periodično sinhronizuje podatke i obezbeđuje da se svaka promena podataka univerzalno ažurira u bazi podataka.

Hijerarhijske baze podataka organizuju podatke modela u strukturi nalik stablu. Skladištenje podataka je format odozgo nadole ili odozdo prema gore i predstavljeno je pomoću odnosa roditelj-dete.

Model mrežne baze podataka rešava potrebu za složenijim odnosima dozvoljavajući svakom detetu da ima više roditelja. Entiteti su organizovani u graf kome se može pristupiti kroz nekoliko putanja.

Objektno orijentisani modeli čuvaju podatke u objektima umesto u redovima i kolonama. Zasnovan je na objektno orijentisanom programiranju (OOP) koje omogućava objektima da imaju članove kao što su polja, svojstva i metode.

Sistemi za upravljanje relacionim bazama podataka (RDBMS) su najpopularniji model podataka zbog svog korisničkog interfejsa. Zasnovan je na normalizaciji podataka u redovima i kolonama tabela. Ovo je izvodljiva opcija kada vam je potreban sistem za skladištenje podataka koji je skalabilan, fleksibilan i sposoban da upravlja puno informacija.

Postoji nekoliko najpopularnijih sistema za upravljanje bazama podataka (SZUBP) koji se široko koriste u industriji. Neki od njih su:

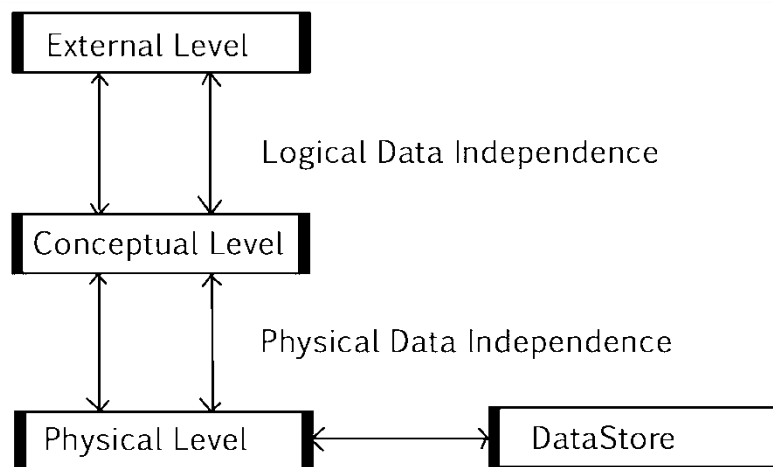
- MySQL - MySQL je open-source SZUBP sistem koji je poznat po svojoj fleksibilnosti, brzini i jednostavnom korišćenju. Široko se koristi za različite vrste aplikacija, od malih web sajtova do velikih korporativnih sistema.
- Microsoft SQL Server - Ovaj SZUBP sistem razvijen je od strane Microsoft-a i često se koristi u Windows okruženjima. Microsoft SQL Server je poznat po svojoj integraciji sa drugim Microsoft proizvodima i podršci za poslovnu inteligenciju.

- Oracle Database - Oracle je jedan od najmoćnijih SZUBP sistema, često korišćen u velikim korporativnim okruženjima. Nudi visoke performanse, napredne funkcionalnosti i snažne alate za upravljanje podacima.
- PostgreSQL - PostgreSQL je open-source SZUBP sistem koji se ističe po svojoj proširivosti i podršci za kompleksne upite. Koristi se u različitim aplikacijama, a posebno je popularan među razvojnim zajednicama.
- MongoDB - MongoDB je NoSQL baza podataka koja se fokusira na rad sa nestrukturiranim podacima. Poznata je po svojoj skalabilnosti i brzini pristupa podacima.

Svaki od ovih SZUBP sistema ima svoje prednosti i specifične karakteristike, a izbor zavisi od potreba projekta, vrste podataka i preferencija korisnika. Fokus ovog rada je **MySQL** sistem za upravljanje relacionim bazama podataka (RDBMS) i njegova interna struktura i organizacija. [8]

3. Interna struktura i organizacija sistema za upravljanje bazama podataka

DBMS ima troslojnu arhitekturu koja deli kompletan sistem na tri međusobno povezana, ali nezavisna modula kao što je prikazano na slici 1.



Slika 1. Troslojna arhitektura DBMS-a [9]

Tri nivoa prisutna u ovoj arhitekturi su:

- Fizički nivo (engl. Physical)
- Konceptualni nivo (engl. Conceptual) – takođe poznat kao logički nivo
- Eksterni nivo (engl. External) – takođe poznat kao aplikacioni nivo

Na fizičkom nivou čuvaju se informacije o lokaciji objekata baze podataka u skladištu podataka. Različiti korisnici DBMS-a nisu svesni lokacija ovih objekata. Jednostavno rečeno, fizički nivo baze podataka opisuje kako se podaci čuvaju u sekundarnim uređajima za skladištenje kao što su diskovi i trake i takođe daje uvid u dodatne detalje o skladištenju. Na najnižem nivou, ovi podaci se čuvaju na eksternim hard diskovima u obliku bitova, a na malo višem nivou, može se reći da se podaci čuvaju u fajlovima i fasciklama. Ovo je najniži nivo u trostepenoj arhitekturi. Takođe je poznat kao unutrašnji nivo. Na fizičkom nivou se takođe govori o tehnikama kompresije i šifrovanja.

Konceptualni nivo je na višem nivou od fizičkog. Takođe je poznat kao logički nivo. Na konceptualnom nivou, podaci su predstavljeni u obliku različitih tabela baze podataka. On opisuje kako se baza podataka čini korisnicima konceptualno i odnose između različitih tabela podataka. Konceptualni nivo ne brine o tome kako se podaci u bazi podataka zapravo čuvaju.

Eksterni nivo je najviši nivo u arhitekturi od tri nivoa i najbliži korisniku. Takođe je poznat kao nivo prikaza ili aplikacioni nivo. Eksterni nivo samo prikazuje relevantan sadržaj baze podataka korisnicima u vidu prikaza i sakriva ostale podatke. Dakle, različiti korisnici mogu da vide bazu podataka kao različit pogled prema njihovim individualnim zahtevima. [9], [10]

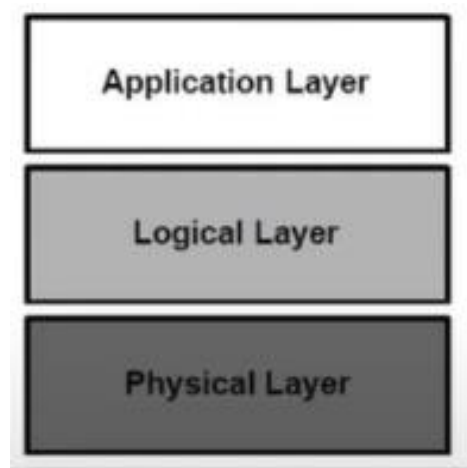
4. MySQL sistem za upravljanje bazama podataka

MySQL je sistem za upravljanje relacionim bazama podataka (engl. Relational Database Management System, RDBMS) koji pokreće server koji pruža višekorisnički pristup brojnim bazama podataka. Radi jako velikom brzinom, pouzdan je i lak za korišćenje. MySQL se često koristi u aplikacijama kao komponenta LAMP steka (Linux, Apache HTTP Server, MySQL i PHP, Perl ili Python). Koriste ga Google, Wikipedia, Facebook, Nokia i Youtube.

MySQL sistem koristi klijent/server arhitekturu. Server je centralni program koji upravlja sadržajem baze podataka, a klijentski programi se povezuju na server da preuzmu ili modifikuju podatke. Kao što je pomenuto u poglavlju 3, sistemi za upravljanje relacionim bazama podataka imaju slojevitú arhitekturu koja ima tri sloja: **fizički nivo**, **konceptualni (logički) nivo** i **eksterni (aplikacioni) nivo**. Ovakvu podelu po slojevima ima i MySQL sistem koja je detaljno objašnjena u sledećim poglavljima. [2]

5. Interna struktura i organizacija MySQL skladišta podataka

Fizička arhitektura MySQL sistema je slojevita i sadrži tri povezane komponente, kao što se vidi na slici 2.



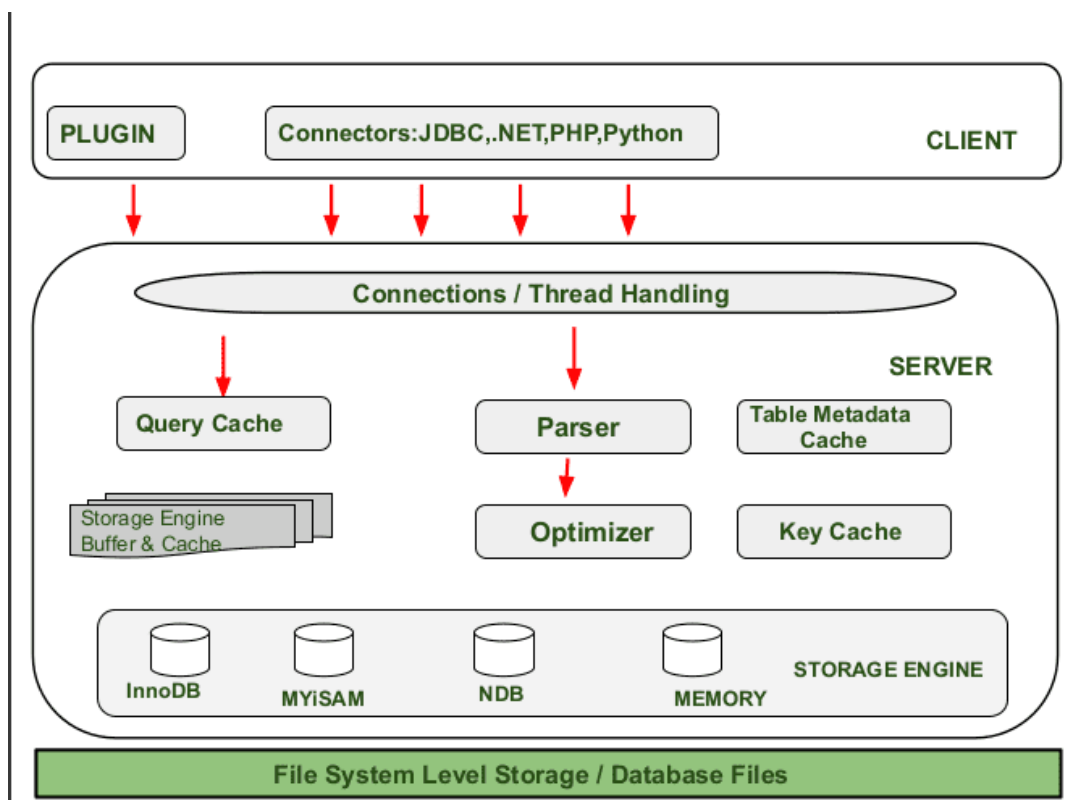
Slika 2. Slojevi arhitekture MySQL sistema [2]

MySQL logička arhitektura bazira se na klijent-server principu. Na slici 3, ispod, može se videti da se na vrhu arhitekture nalazi klijent, koji predstavlja uslugu koja omogućava povezivanje sa MySQL serverom. Ovaj klijent može biti bilo koji plugin ili neki od konektora poput JDBC, .NET, JSON konektora, ili čak aplikacija koja pokušava uspostaviti vezu sa MySQL serverom. Ova raznolikost omogućava fleksibilnost i prilagodljivost sistema prema različitim potrebama korisnika.

Nakon klijenta, sledeći sloj odnosi se na sam server koji deluje kao posrednik i obavlja celokupnu logiku sistema. Na slici 3 može se videti da se na server strani nalaze konektori, keš upita, parser, optimizator, keš tabele metapodataka, keš ključevi i mašine za skladištenje kao što su InnoDB, MyISAM, NDB, i drugi. MySQL koristi server proces *mysqld*, koji predstavlja program u pozadini koji efikasno upravlja bazom podataka. Kada se klijent poveže na server, svaki klijent predstavlja jednu nit, a svakom klijentu je dodeljena jedna konekcija sa odgovarajućom niti. Mysqld je višenitni proces, što znači da, ukoliko je potrebno obezbediti 10 konekcija za 10 klijenata, mysqld mora pružiti različite niti za svaku od njih. Ova jedan-na-jedan relacija između klijenta i MySQL servera obezbeđuje efikasno upravljanje vezama i resursima.

Na dnu ove arhitekture nalazi se sloj za skladištenje podataka, gde se informacije trajno čuvaju i organizuju. Ovaj sloj čini osnovu za sigurno čuvanje podataka i pruža različite

tehnologije skladištenja, poput InnoDB, MyISAM, NDB, koje se mogu optimalno prilagoditi potrebama aplikacija. [2]



Slika 3. Klijent-server arhitektura MySQL sistema [1]

5.1. Aplikacioni sloj

Aplikacioni sloj, takođe poznat kao konekcioni sloj ili klijentski kraj MySQL strukture, predstavlja ključnu komponentu koja omogućava interakciju krajnjih korisnika sa sistemom baze podataka. Ova ključna tačka omogućava korisnicima slanje više MySQL komandi serveru, kako putem komandne linije, tako i putem grafičkog korisničkog interfejsa. Ovaj sloj igra ključnu ulogu u olakšavanju komunikacije između korisnika i baze podataka, omogućavajući im da efikasno upravljaju podacima.

Prilikom slanja validnih komandi, sistem prikazuje relevantan izlaz na ekranu, pružajući korisnicima potrebne informacije. U slučaju neispravnog podnošenja komande, sistem reaguje slanjem obaveštenja o grešci kao povratnu informaciju na ekran. Ova transparentnost u

komunikaciji korisnika sa bazom podataka obezbeđuje korisničko iskustvo koje je intuitivno i efikasno.

Aplikacioni sloj obuhvata nekoliko ključnih usluga, od kojih su najvažnije:

- **Rukovanje vezom** - Ova usluga omogućava uspostavljanje i održavanje veze između klijenta i MySQL servera. Efikasno rukovanje vezom ključno je za stabilnu i pouzdanu komunikaciju.
- **Autentifikacija** - Kroz ovu uslugu, sistem potvrđuje identitet korisnika, pružajući siguran pristup i obezbeđujući da samo ovlašćeni korisnici imaju pristup određenim resursima.
- **Bezbednost** - Bezbednosne aspekte aplikacionog sloja čine mehanizmi koji štite podatke od neovlašćenog pristupa i obezbeđuju integritet tokom komunikacije između klijenta i servera.

5.1.1. Rukovanje vezom

Kada klijent uputi zahtev, server ga potvrđuje, uspostavlja vezu koja omogućava klijentu izvršavanje niza zahteva. Ova sposobnost, poznata kao **rukovanje vezom**, predstavlja karakterističnu funkcionalnost klijentske strane MySQL strukture. Prilikom uspostavljanja veze sa serverom, svaki klijent dobija svoju posebnu nit koja je posvećena upravljanju tom vezom. Svi upiti koje klijent šalje izvršavaju se unutar ove namenske niti, pružajući izolaciju i efikasno rukovanje klijentskim zahtevima.

Server kešira ove niti, eliminirajući potrebu za konstantnim kreiranjem i uništavanjem niti za svaku novu vezu. Ova tehnika keširanja niti omogućava efikasno upravljanje resursima i smanjenje opterećenja servera, čime se poboljšava ukupna performansa sistema.

Ovaj integrisani pristup rukovanja vezom dodatno optimizuje komunikaciju između klijenta i servera. Tako se postiže brža i efikasnija razmena podataka, pružajući korisnicima pouzdan i odzivan sistem za upravljanje bazama podataka.

Aplikacioni sloj funkcioniše kao prijemna tačka za veze iz aplikacija putem nekoliko različitih komunikacionih protokola, koji su navedeni u nastavku i prikazani na slici 4:

- TCP/IP
- UNIX soketa
- Zajednička memorija
- Imenovane cevi

Ova raznovrsnost komunikacionih protokola omogućava prilagodljivost i kompatibilnost aplikacionog sloja, omogućavajući mu da efikasno prihvata veze iz različitih izvora i okruženja. Ova interoperabilnost pruža širok spektar opcija za komunikaciju sa MySQL sistemom, prilagođavajući se specifičnim zahtevima aplikacija i operativnog okruženja. [4]

Protocol	Types of Connections	Supported Operating Systems
TCP/IP	Local, remote	All
UNIX socket file	Local only	UNIX only
Shared memory	Local only	Windows only
Named pipes	Local only	Windows only

Slika 4. Komunikacioni protokoli [4]

TCP/IP

TCP/IP (Protokol za kontrolu prenosa/Internet protokol) omogućava pouzdanu i efikasnu globalnu komunikaciju između klijenta i servera na različitim nivoima mreže. S druge strane, ostali prethodno pomenuti protokoli, uključujući UNIX sokete, zajedničku memoriju i imenovane cevi, podržavaju isključivo lokalne veze. Ova lokalna ograničenja znače da su efikasni samo kada klijent i server funkcionišu na istoj fizičkoj mašini ili uređaju.

Aplikacioni sloj, kao posrednik u ovim komunikacionim procesima, održava jednu nit po svakoj vezi kako bi efikasno upravljao izvršavanjem SQL upita. Svaka nit ima odgovornost za upravljanje specifičnom vezom, pružajući izolaciju između različitih veza i unapređujući stabilnost i performanse sistema.

Pre nego što veza započne slanje SQL upita, neophodno je sprovesti proces potvrde koji uključuje verifikaciju korisničkog imena, lozinke i klijentskog hosta. Ova faza garantuje pristup samo ovlašćenim korisnicima, pružajući visok nivo bezbednosti i autentičnosti u interakciji između klijenta i servera. Ovaj pažljivo postavljen proces potvrde predstavlja nerazdvojni deo mehanizama sigurnosti na aplikacionom sloju.

MySQL koristi DNS (Domain Naming System) za razrešavanje imena hostova povezanih putem TCP/IP protokola, čuvajući ih u keš memoriji hosta.

TCP/IP se koristi za povezivanje hostova na Internetu i postaje standard u Linux operativnom sistemu, dok je takođe preporučen za veze u Windows okruženju. [4]

Unix socket

UNIX socket predstavlja efikasan mehanizam međuprocesne komunikacije, omogućavajući dvosmernu vezu između procesa na istom uređaju. Ovaj tip veze, posebno koristan u Linux okruženju, eliminiše potrebu za mrežnom infrastrukturom, čime se smanjuje latencija i poboljšavaju ukupne performanse sistema.

Karakteristična prednost UNIX socketa ogleda se u sposobnosti ostvarivanja direktnih veza između lokalnih procesa, pritom zahtevajući fizičku datoteku na lokalnom sistemu kao identifikator. Ovo čini UNIX socket izuzetnim izborom za efikasnu i brzu međuprocesnu komunikaciju.

Posebno, kada se koristi u kontekstu MySQL sistema, UNIX socket može poslužiti kao efikasan način za lokalnu konekciju između aplikacija i MySQL servera, čime se dodatno unapređuju performanse i odziv u sistemu baze podataka. [4]

Zajednička memorija

Zajednička memorija efikasno omogućava prenos podataka između programa. U ovom postupku, jedan program kreira memorijski segment, a drugi procesi, uz odobrenje, može pristupiti tom segmentu. Ovaj "pasivni" režim operativnog sistema Windows funkcioniše unutar jedne mašine, ali podrazumevano je onemogućen. Da bi se aktivirala deljena memorija, server se pokreće sa opcijom *–shared-memory*. Ovaj pristup optimizuje efikasnost i resurse, pružajući programima mogućnost bezbedne razmene informacija. Time se unapređuje efikasnost i fleksibilnost komunikacije između programa na Windows operativnom sistemu. [4]

Imenovane cevi

Imenovane cevi predstavljaju preferirani način komunikacije u klijent/server arhitekturi, delujući slično kao soketi. Ove cevi omogućavaju operacije čitanja i pisanja, pritom nudeći "pasivni" režim koji je eksplicitno usmeren ka serverskim aplikacijama. Bitno je napomenuti da ovaj protokol funkcioniše isključivo unutar Windows mašine, pružajući lokalnu komunikaciju.

Podrazumevano, imenovane cevi su onemogućene iz bezbednosnih razloga. Mogu se omogućiti pokretanjem servera sa dodatnom opcijom *–enable-named-pipe*. Ovom konfiguracijom, sistem postaje sposoban za efikasno uspostavljanje komunikacije putem imenovanih cevi, što je posebno korisno u lokalnim okruženjima i za serverske aplikacije koje zahtevaju ovu vrstu komunikacije. [4]

5.1.2. Autentifikacija

Autentifikacija u MySQL okruženju je proces koji se odvija prilikom svakog povezivanja korisnika, a ključni je aspekt sigurnosti. Ova procedura ima dva osnovna elementa - jedan deo se odvija na klijentu, dok drugi ima svoje mesto na serveru.

```
C:\Program Files\MySQL\MySQL Server\bin>mysql -u root -h localhost -p
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
```

Slika 5. Primer autentifikacije

Kada klijent želi uspostaviti vezu sa MySQL serverom, prvi korak autentifikacije dešava se na strani klijenta. Klijent pruža serveru korisničko ime i lozinku, čime inicira proces provere identiteta. Naredba sa slike 5 koristi se za uspostavljanje veze s lokalnim MySQL serverom koristeći korisničko ime *root* i host *localhost*. Nakon što se pokrene, korisnik će biti upitan za unos lozinke za korisnika *root*. Nakon uspešne autentifikacije, korisnik može pristupiti i upravljati MySQL bazom podataka.



```
1 • CREATE USER 'john'@'remote_host' IDENTIFIED BY 'password';
2 • GRANT ALL PRIVILEGES ON *.* TO 'john'@'remote_host' WITH GRANT OPTION;
3 • FLUSH PRIVILEGES;
```

Slika 6. Kreiranje novog korisnika

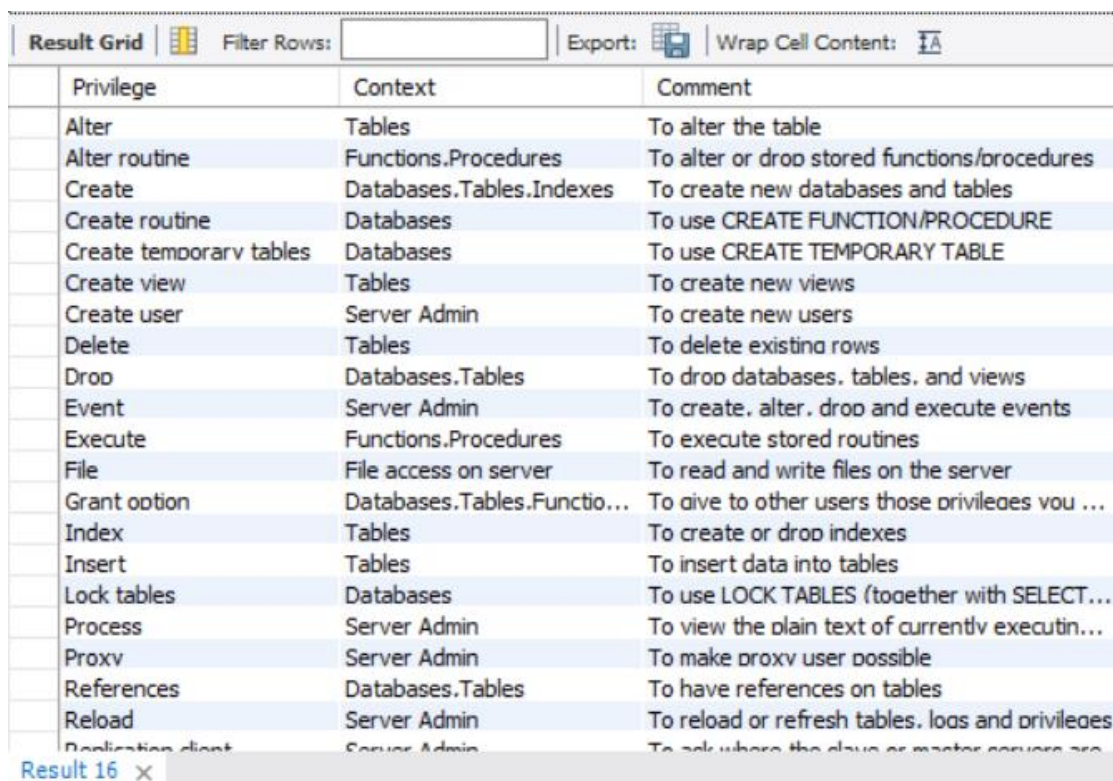
S druge strane, drugi deo autentifikacije odvija se na strani servera. Na primeru SQL komande `CREATE USER 'john'@'remote_host' IDENTIFIED BY 'password'`, na slici 6, kreiranje korisničkog naloga vrši se na serveru. Server proverava ovu informaciju i, kada se korisnik "john" poveže sa udaljenog hosta, vrši se potvrda autentifikacije. Ova verifikacija uključuje proveru podudaranja korisničkog imena, hosta i lozinke, što serveru omogućava precizno identifikovanje izvora konekcije.

Praktičan primer autentifikacije u MySQL-u, poput omogućavanja pristupa korisniku "john" sa udaljenog hosta, direktno ilustruje kako se autentifikacija koristi za uspostavljanje kontrolisane i sigurne konekcije. Ovaj dinamičan proces integriše i informacije sa obe strane - klijenta i servera, kako bi osigurao celovitost i sigurnost pristupa podacima.

5.1.3. Bezbednost

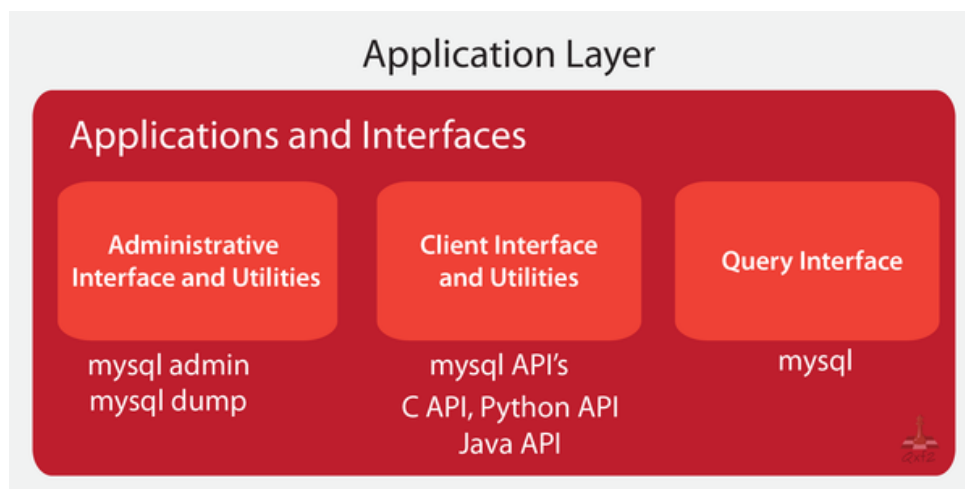
Bezbednost na aplikacionom sloju MySQL sistema obuhvata mehanizme dizajnirane za zaštitu podataka od neovlašćenog pristupa i osiguravanje integriteta tokom komunikacije između klijenta i servera. Kada korisnici uspešno prođu autentifikaciju, povezuju se na MySQL server, gde server dalje proverava privilegije korisnika, neophodne za postavljanje specifičnih upita prema MySQL serveru.

Privilegije korisnika mogu se pogledati putem SQL upita `SHOW PRIVILEGES;` nakon čijeg izvršenja se dobija izlaz koji prikazuje Privilegije (privilege), Kontekst (context) i Komentar (comment) kao na slici 7. Polje privilegije sadrži nazive različitih privilegija koje korisnik može imati, odnosno dozvole koje kontrolišu šta korisnici mogu raditi u bazi podataka. Polje kontekst označava kontekst u kojem je privilegija primenljiva. Komentar opisuje šta privilegija omogućava korisniku.



Privilege	Context	Comment
Alter	Tables	To alter the table
Alter routine	Functions.Procedures	To alter or drop stored functions/procedures
Create	Databases.Tables.Indexes	To create new databases and tables
Create routine	Databases	To use CREATE FUNCTION/PROCEDURE
Create temporary tables	Databases	To use CREATE TEMPORARY TABLE
Create view	Tables	To create new views
Create user	Server Admin	To create new users
Delete	Tables	To delete existing rows
Drop	Databases.Tables	To drop databases, tables, and views
Event	Server Admin	To create, alter, drop and execute events
Execute	Functions.Procedures	To execute stored routines
File	File access on server	To read and write files on the server
Grant option	Databases.Tables.Functions	To give to other users those privileges you ...
Index	Tables	To create or drop indexes
Insert	Tables	To insert data into tables
Lock tables	Databases	To use LOCK TABLES (together with SELECT ...)
Process	Server Admin	To view the plain text of currently executing queries
Proxy	Server Admin	To make proxy user possible
References	Databases.Tables	To have references on tables
Reload	Server Admin	To reload or refresh tables, logs and privileges
Replication client	Server Admin	To ask where the slave or master servers are

Slika 7. Prikaz privilegija korisnika



Slika 8. MySQL interfejsi na aplikacionom sloju [3]

Postoje tri glavne komponente u ovom sloju, a to su administratori, klijenti, korisnici upita kao što je prikazano na slici 8.

Administratori, kao prva ključna komponenta ovog sloja, koriste različite administrativne interfejse i alate poput *mysqladmin* za zadatke kao što su gašenje servera, kreiranje ili izbacivanje baza podataka, ili *mysqldump* za pravljenje rezervnih kopija baza podataka.

Klijenti, kao druga komponenta, komuniciraju sa MySQL-om putem različitih interfejsa i programa, uključujući MySQL API. Ovi klijentski programi izvršavaju se iz komandne linije i obavljaju različite zadatke, kao što je izdavanje upita ili administracija servera. [3]

MySQL komunikacija klijent/server nije ograničena na okruženja u kojima svi računari pokreću isti operativni sistem. Klijentski programi mogu se povezati sa serverom na istom ili drugom hostu, čak i u situacijama gde računari pokreću različite operativne sisteme. Postoji niz klijentskih programa koji se izvršavaju iz komandne linije, svaki sa specifičnom funkcionalnošću poput izdavanja upita, administracije servera, provere integriteta tabela, kreiranja rezervnih kopija itd. Ovi programi se izvršavaju iz komandne linije:

```
shell> mysql [options]
```

Bezbednost na aplikacionom sloju MySQL sistema je esencijalna za očuvanje integriteta podataka i sprečavanje neovlašćenog pristupa. Komponente ovog sloja - administratori, klijenti i korisnici upita - zajedno čine kompleksan ekosistem koji obezbeđuje kontrolisanu i sigurnu interakciju između korisnika i MySQL servera. [4]

5.2. Logički sloj

Drugi sloj MySQL arhitekture, često nazivan i *mozak MySQL arhitekture*, predstavlja ključni entitet koji kontrolše sve logičke funkcionalnosti MySQL sistema za upravljanje relacionim bazama podataka. Ovaj ključni sloj odigrava ključnu ulogu u interpretaciji i izvršavanju zahteva koji dolaze od klijenata, obezbeđujući efikasnu manipulaciju podacima.

Svaki segment MySQL servera u okviru ovog sloja doprinosi integrisanom funkcionisanju sistema. Neke od ključnih podkomponenata u ovom sloju su:

- Komponenta za upravljanje nitima - ova komponenta je odgovorna za upravljanje i kontrolu niti koje se koriste za svaku pojedinačnu vezu sa klijentom. Višenitna priroda MySQL servera omogućava efikasno rukovanje višestrukim klijentskim konekcijama.
- Parser ima ključnu ulogu u razumevanju SQL upita koje klijent šalje serveru. Ovde se vrši analiza i sintaksa upita kako bi se pripremili za dalju obradu.
- Optimizator ima zadatak da optimizuje izvršenje upita, birajući najefikasniji put između dostupnih opcija. Ovo doprinosi poboljšanju performansi sistema.
- Keš upita čuva rezultate prethodno izvršenih upita, što omogućava brži odgovor na identične ili slične zahteve u budućnosti. To doprinosi efikasnosti sistema tako što smanjuje potrebu za ponovnim izvršavanjem istih upita.
- Bafer i keš komponenta čuva privremene rezultate operacija i optimizuje pristup podacima, smanjujući često čitanje sa diskova.
- Keš metapodataka tabele komponenta čuva metapodatke tabele, čime se smanjuje potreba za čestim pristupom samim tabelama tokom izvršavanja upita.
- Keš ključeva doprinosi ubrzanju pristupa podacima čuvanjem često korišćenih ključeva u memoriji.

Ovi elementi zajedno čine centralni deo MySQL sistema, obezbeđujući efikasno i optimizovano rukovođenje podacima u okviru relacionih baza podataka. Kada klijent šalje zahteve serveru, svaki od ovih segmenata doprinosi brzom i preciznoj obradi instrukcija, čime se postiže optimalna funkcionalnost sistema.

Komponenta za upravljanje nitima

Komponenta za upravljanje nitima, jedna od ključnih podkomponenti MySQL servera koje odigrava ključnu ulogu u procesu pružanja efikasnog upravljanja podacima u MySQL sistemu.

Kada klijent uspešno pošalje zahtev, server brzo identifikuje i prihvata taj zahtev, omogućavajući postizanje povezivanja sa klijentom. Ovaj tip veze, poznat kao nit, igra ključnu ulogu u organizaciji podataka unutar MySQL sistema. Nit čini osnovnu strukturu za efikasno rukovođenje podacima tokom različitih operacija.

Sama serverska arhitektura poseduje ključnu komponentu, poznatu kao modul za rukovanje nitima, koji ima značajnu funkcionalnost u upravljanju svakom niti. Ovaj proces omogućava optimalnu kontrolu i koordinaciju niti, garantujući efikasno i koherentno rukovođenje podacima. Osim toga, ovaj ključni modul efikasno upravlja i upitima koje izvršava nit na strani klijenta.

Pored osnovne uloge omogućavanja povezivanja i komunikacije između klijenta i servera, nit u MySQL arhitekturi igra ključnu ulogu u omogućavanju paralelnog izvršavanja različitih zahteva. Ova sposobnost doprinosi značajnom povećanju efikasnosti sistema, omogućavajući simultano rukovođenje više zahteva bez gubitka performansi.

Komponenta za upravljanje nitima u MySQL arhitekturi predstavlja vitalni deo serverske strukture, pružajući osnovu za brzo, sinhronizovano i efikasno rukovođenje podacima tokom procesa izvršavanja upita. [1]

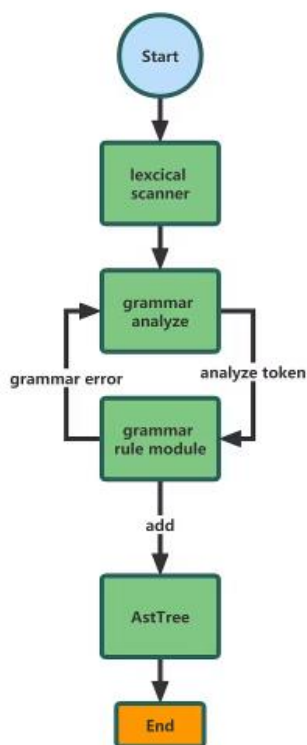
Parser

Parser kao ključna komponenta MySQL sistema, odigrava presudnu ulogu u obradi SQL upita nakon što klijent izda zahtev. Kada se novi zahtev pojavi, stvara se nova nit, a SQL upit se prosleđuje parseru radi temeljne sintaksičke provere, sa ciljem identifikacije i eventualnog odbijanja upita zbog detektovanih grešaka.

MySQL parser se temelji na moćnoj Lek-IACC skripti koja se kompajlira pomoću alata poput Bison-a. Proces raščlanjivanja podataka unutar baze se sprovodi kroz leksičku analizu, koja razbija tekst upita na različite tokene. Parser, kao softverska komponenta, generiše strukturu podataka u obliku stabla raščlanjivanja na osnovu ulaznog upita.

Ovaj proces uključuje leksički skener, zadužen za pretvaranje upita u tokene, i modul gramatičkih pravila koji traži odgovarajuće SQL gramatičke obrasce i izvršava povezani kod. Na kraju, parser generiše stablo raščlanjivanja, koje zatim optimizator može iskoristiti u daljem procesu.

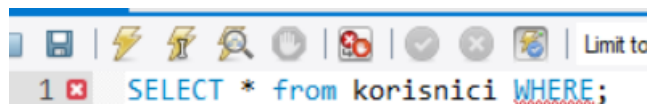
Važno je napomenuti da se, za razliku od nekih parsera koji prevode tekstualnu reprezentaciju upita u bajt kod, MySQL parser odlikuje time što direktno prevodi upit u interne međusobno povezane C++/C strukture unutar programske memorije. Ovaj pristup donosi određene prednosti u efikasnosti i performansama obrade upita. [1], [5], [7]



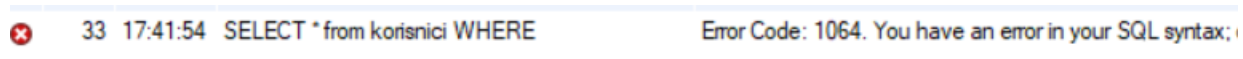
Slika 9. Parser [7]

Slika 9 prikazuje ključni deo MySQL parsera, koji se sastoji od dva osnovna elementa: **leksičkog skenera** i **modula gramatičkih pravila**. Ova dva dela usko sarađuju kako bi efikasno obradili SQL upit koji stiže od strane klijenta.

Leksički skener ima zadatak da razbije SQL izjavu na tokene, čime se formira osnovna struktura podataka koja će dalje biti analizirana. Sa druge strane, modul gramatičkih pravila proverava da li ovi tokeni zadovoljavaju MySQL gramatička pravila. Ukoliko se uoče nepravilnosti ili greške u sintaksi upita, server će generisati odgovarajuću SQL grešku, pružajući korisnicima jasnu povratnu informaciju o problemu. Primer toga je prikazan na slici 10, gde SQL upit " *SELECT * FROM korisnici WHERE;*" rezultira SQL greškom 1064, kao na slici 11 .

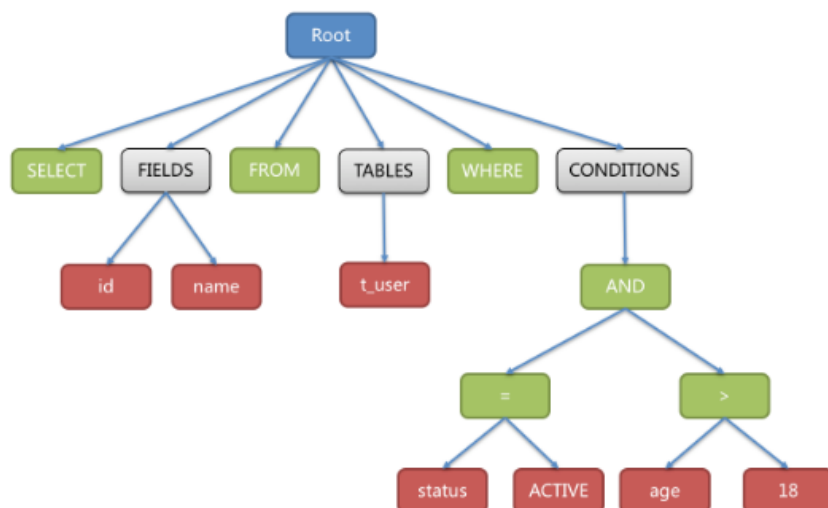


Slika 10. Primer sintaksne greske u SQL upitu



Slika 11. Error kod 1064

Nakon uspešnog leksičkog skeniranja, dobija se **stablo raščlanjivanja** (Slika 12). Stablo raščlanjivanja predstavlja apstraktnu strukturu podataka koja odražava hijerarhijski raspored tokena u okviru SQL upita. Ovo stablo je ključno za proces dalje optimizacije, jer omogućava serveru da efikasno manipuliše i organizuje tokene u skladu sa gramatičkim pravilima MySQL sistema.



Slika 12. Stablo raščlanjivanja [7]

Optimizator

Optimizator, ključna komponenta MySQL arhitekture, predstavlja kritičan korak u procesu izvršavanja upita. Nakon uspešnog raščlanjivanja SQL upita, sistem primenjuje raznovrsne tehnike optimizacije kako bi obezbedio efikasnost izvršavanja. Ovaj ključni deo arhitekture pruža MySQL sistemu sposobnost da predviđa i odabere najoptimalniji plan izvršenja.

MySQL se oslanja na pristup optimizaciji zasnovan na troškovima, gde pokušava predvideti troškove različitih planova izvršenja i odabrati najjeftiniji. Inicijalno, jedinica troška

bila je jedna proizvoljna pročitana stranica sa podacima od 4 KB, ali tokom vremena, ovaj proces postao je sofisticiraniji, uključujući faktore poput troškova poređenja klauzula WHERE. [1]

Prvi korak u optimizatoru obuhvata proveru postojanja tabela i kontrolu pristupa od strane korisnika. Ukoliko se uoče greške, korisnicima se vraća odgovarajuća poruka o grešci. Kada se identifikuju ispravne tabele, one se otvaraju, a primenjuju se odgovarajuća zaključavanja radi očuvanja konkurentnosti. Ova faza uključuje i procenu WHERE uslova upita, čiji se rezultati koriste za pripremu privremenih tabela za naredne korake.

Ako upit sadrži UNION operatore, optimizator izvršava SELECT delove svih naredbi u petlji pre nego što nastavi sa izvršavanjem. Sledeći korak obuhvata izvršavanje projekcija, slično ograničenim delovima, čuvajući međurezultate kao privremene tabele i zadržavajući samo one attribute navedene u specifikaciji kolona u SELECT naredbama.

Konačno, optimizator analizira strukturu za sve JOIN uslove i poziva odgovarajuće metode za optimizaciju. U ovom koraku, upit se optimizuje procenom izraza i eliminacijom uslova koji vode do mrtvih grana ili uvek tačnih/lažnih uslova. Pridruživanje optimizacije posebno se ističe, budući da se spajanja smatraju najskupljim i najzahtevnijim od svih relacionih operatora.

Kada je optimizacija pridruživanja završena, optimizator usmerava upit do odgovarajuće metode biblioteke za izvršenje, čime se kompletira ključni proces optimizacije u MySQL sistemu. [5]

Keš upita

Keš upita pruža mehanizam efikasnog čuvanja rezultata često izvršavanih upita. Pre nego što se upit podvrgne procesu raščlanjivanja, server proverava keš upita, koji specifično čuva SELECT izraze i pripadajuće skupove rezultata. Ova strategija omogućava serveru da preskoči raščlanjivanje, optimizaciju i izvršavanje upita ako pronađe odgovarajući upit u kešu, umesto da ponovo izvršava sve korake.

MySQL inicijalizuje keš upita i dodeljuje mu određenu količinu memorije kada server pokrene, a vrednost se preuzima iz promenljive *query_cache_size*. Ako se ova promenljiva ažurira ili postavi na drugu vrednost, svi keširani upiti se brišu, keš memorija se razbija na određene veličine i ponovo se pokreće.

Prilikom primanja upita, MySQL proverava keš upita da utvrdi da li je sličan upit već izvršen. Upiti i rezultati se čuvaju u kešu upita kao parovi ključ-vrednost. Kada se ključevi

podudaraju, rezultati se brzo dobijaju iz keša, bez potrebe za ponovnim pristupanjem bazi podataka.

Iako je keš upita često koristan, njegova efikasnost može opasti u situacijama gde se podaci u tabeli često menjaju, što uzrokuje brisanje svih keširanih upita. U takvim slučajevima, preporučuje se njegovo korišćenje na statičnim tabelama. Takođe, važno je napomenuti da ključ (izraz za odabir) mora ostati nepromenjen kako bi keš bio efikasan.

Ukoliko se neko odluči za korišćenje keš upita, preporučuje se postavljanje MySQL promenljive *query_cache_type* na DEMAND, čime se obezbeđuje kontrolisano i prilagođeno korišćenje keša. Ova postavka se vrši u MySQL konfiguracionom fajlu (mysql.cnf) i može se kombinovati sa dodavanjem 'SQL_CACHE' u izraz upita, kao na slici 13, kako bi se aktivirao keš. [1], [7]



Slika 13. Sql_cache

Keš upita (query cache) zastareo od verzije MySQL 5.7.20 i uklonjen je od MySQL 8.0 verzije.

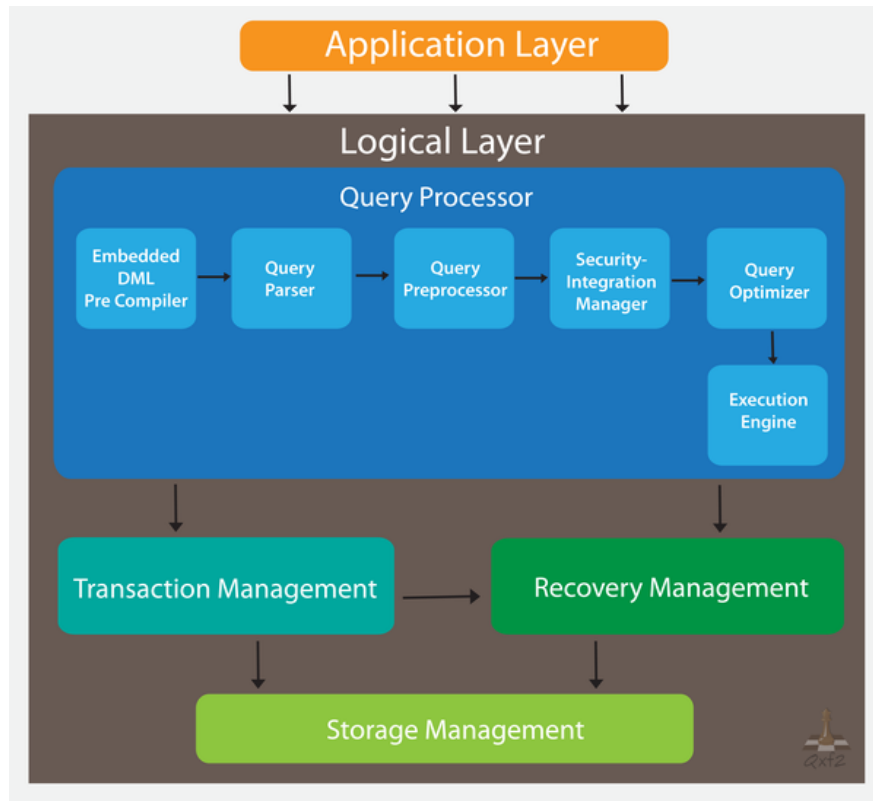
Keš metapodataka tabele

Keš metapodataka tabele predstavlja odvojeni segment memorije koji čuva informacije o objektima, bazama podataka i indeksima u MySQL sistemu. Veličina ovog keša direktno zavisi od broja otvorenih baza podataka, indeksa i objekata. Povećanje broja otvorenih indeksa, objekata ili baza podataka rezultuje i povećanjem veličine keša metapodataka.

Ova komponenta igra ključnu ulogu u efikasnom upravljanju metapodacima, što omogućava brz pristup informacijama o strukturi baza podataka i indeksima. Povećanjem veličine keša metapodataka, sistem može efikasnije pratiti i upravljati složenim strukturama podataka, poboljšavajući tako ukupnu performansu sistema.

Keš ključeva

Keš ključeva predstavlja ključnu podkomponentu keša koja efikasno čuva blokove indeksa na koje se često pristupa. Ova komponenta igra ključnu ulogu u optimizaciji performansi sistema za upravljanje bazama podataka, smanjujući disk I/O operacije zahvaljujući brzom pristupu memoriji, koja je znatno brža od čvrstih diskova. Ovim pristupom postiže se ubrzanje procesa čitanja indeksa, čime se poboljšava ukupna efikasnost sistema. [1]



Slika 14. Logički sloj i njegove podkomponente [3]

Logički sloj MySQL arhitekture obuhvata nekoliko ključnih podsistema, čime se osigurava efikasno upravljanje podacima. Na slici 14, mogu se primetiti sledeći podsistemi:

- Obrada upita (engl. Query Processor)
- Upravljanje transakcijama (engl. Transaction Management)
- Menadžment oporavka (engl. Recovery Management)
- Menadžment skladištenja (engl. Storage Management)

Ovi podsistemi čine temelj za različite funkcionalnosti MySQL sistema, uključujući efikasnu obradu upita, sigurno upravljanje transakcijama, pouzdan menadžment oporavka i optimalno upravljanje skladištem podataka. Svaki od njih igra ključnu ulogu u obezbeđivanju stabilnosti, sigurnosti i performansi MySQL sistema.

Obrada upita (Query Processor) je odgovorna za analizu SQL upita i određivanje optimalnog izvršnog plana. Upravljanje transakcijama (Transaction Management) se bavi transakcijama, omogućavajući ACID karakteristike (Atomicity, Consistency, Isolation, Durability) za sigurno izvršavanje transakcija. Menadžment oporavka (Recovery Management) se bavi oporavkom sistema u slučaju pada ili neuspeha. Menadžment skladištenja (Storage Management) se fokusira na efikasno upravljanje fizičkim skladištenjem podataka na disku.

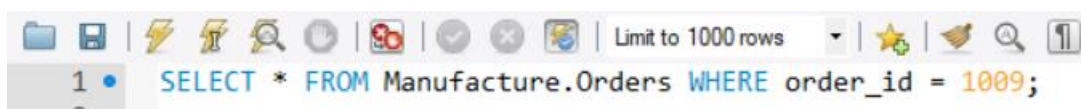
5.2.1. Podsistem obrade upita

Procesor upita (Query Processor) predstavlja ključni deo logičkog sloja MySQL arhitekture, sastavljen od niza komponenata koje zajednički omogućavaju efikasnu i bezbednu obradu upita. Ove komponente mogu se videti na slici 14, a uključuju:

- Embedded DML pre-compiler i DDL compiler - Ove komponente se bave prekompilacijom i kompilacijom Data Manipulation Language (DML) i Data Definition Language (DDL) upita, obezbeđujući pravilnu sintaksu i strukturu upita pre nego što se proslede dalje na obradu.
- Query parser - Parser se bavi analizom sintakse upita, razbijajući ga na pojedinačne elemente radi dalje obrade. Ovo osigurava ispravnu interpretaciju upita u skladu sa MySQL gramatičkim pravilima.
- Query preprocessor - Preprocessor vrši predobradu upita, pripremajući ih za naredne faze procesiranja. Ovo može uključivati provere pristupa, pretraživanje keš memorije i druge korake pripreme.
- Security/Integration Manager - Komponenta za bezbednost i integraciju upravlja sigurnosnim aspektima obrade upita i integriše ih u širi kontekst sistema. To uključuje provere prava pristupa i integraciju sa sigurnosnim mehanizmima sistema.
- Query Optimizer - Optimizator upita ima ključnu ulogu u odabiru optimalnog plana izvršenja upita. Koristi se različitim tehnikama optimizacije kako bi se minimiziralo vreme izvršenja upita i optimizovala efikasnost.
- Execution Engine - Izvršni mehanizam bavi se konkretnom implementacijom izabranog plana izvršenja. Ovo uključuje izvršavanje upita na stvarnim podacima i manipulaciju rezultatima.

Klijent se povezuje na logički sloj preko aplikativnog sloja, gde Query Processing komponenta preuzima upit od klijenta. Nakon obrade u smislu komunikacije i optimizacije, upit se prosleđuje Transaction Management komponenti, što predstavlja ključni korak u upravljanju transakcijama MySQL sistema.

Tok obrade upita objašnjen na primeru SELECT naredbe



Slika 15. Primer

Prilikom izvršavanja sa slike 15 u MySQL arhitekturi, proces prolazi kroz niz ključnih faza. Kada klijent šalje zahtev, ugrađena DML komponenta prepoznaje SQL izraze i prevodi ih u odgovarajuće SQL izraze kako bi osigurala pravilnu sintaksu. Nakon uspostavljanja klijentske sesije, kreira se MySQL THREAD za rukovanje komandama i sesijama, a upit se šalje MySQL serveru (mysqld procesu).

U MySQL 8.0, keš upita se koristi samo ako je upit već izvršen i rezultat je sačuvan u kešu, inače se nastavlja s izvršavanjem upita. Ako upit nije u kešu, ulazi se u fazu parsera koji registruje klauzule upita, uključujući SELECT, FROM i WHERE, i kreira strukturu stabla raščlanjivanja na osnovu SQL izraza.

Pretprocesor upita proverava SQL sintaksu i semantiku, uključujući vrednost *order_id*, kako bi utvrdio ispravnost upita. Menadžer bezbednosti i integracije proverava dozvole korisnika za pristup objektima, a zatim se upit prenosi optimizatoru upita. Optimizator analizira indekse u tabeli *Orders* i određuje plan izvršenja za brže preuzimanje podataka.

Kada se odobri pristup, MySQL kreira plan izvršenja i prenosi ga sloju mehanizma za skladištenje (Storage Engine). U ovom sloju, izvršava se upit koristeći indeks na *order_id* za efikasno lociranje i preuzimanje podataka. Rezultati se zatim šalju klijentu, završavajući proces efikasne i optimizovane obrade upita. Ovaj pristup obezbeđuje tačnost u rezultatima upita i istovremeno optimizuje brzinu pretrage i hvatanja podataka, posebno fokusirajući se na klauzulu *WHERE order_id = 1009* kako bi se postigla efikasnost izvršavanja u kontekstu ovog specifičnog uslova. [3]

5.2.2. Upravljanje transakcijama

Upravljanje transakcijama u okviru MySQL sistema igra ključnu ulogu u održavanju integriteta podataka i obezbeđivanju doslednosti u bazi podataka. Transakcije mogu biti pojedinačne ili grupisane, pružajući fleksibilnost u radu s podacima.

Kada se govori o pojedinačnim transakcijama, to obično podrazumeva izvršavanje određenih radnji unutar jedne transakcije. Međutim, može se dogoditi da jedna transakcija poziva izvršenje druge transakcije, stvarajući niz povezanih transakcija. Na primer, prva transakcija može inicirati izvršenje druge, koja zatim izvršava sledeću transakciju, a rezultat se na kraju vraća prvoj transakciji.

Transaction Management preuzima odgovornost za praćenje ovakvih transakcionih nizova. Kroz ovu komponentu, MySQL sistem održava redosled izvršavanja transakcija, beležeći svaku pojedinačnu radnju i osiguravajući doslednost rezultata. Ova evidencija je od suštinskog

značaja kako bi se omogućilo efikasno vođenje transakcija i omogućilo povratne informacije između njih.

U grupisanim transakcijama, više radnji se grupiše zajedno kao jedna transakcija. Ovaj pristup je koristan kada želimo obezbediti da se sve radnje izvrše kao jedna celina - ako jedna radnja ne uspe, cela transakcija se poništava.

Uz sve ovo, Transaction Management vodi računa o konceptu "commit" i "rollback". Kada je transakcija uspešno izvršena, njen rezultat se "commituje", čime se promene trajno primenjuju na bazi podataka. S druge strane, ako se u transakciji pojavi problem ili greška, moguće je izvršiti "rollback", što znači poništiti sve promene i vratiti bazu podataka u prethodno stanje.

Upravljanje transakcijama u MySQL arhitekturi ne samo da pruža organizaciju i evidenciju za niz povezanih transakcija, već takođe omogućava doslednost i trajnost promena u bazi podataka, čime se obezbeđuje pouzdanost i sigurnost podataka.

5.2.3. Menadžemnt oporavka

Recovery Management igra presudnu ulogu u održavanju stabilnosti baze podataka i obezbeđivanju sposobnosti oporavka u slučaju neplaniranih prekida sistema.

Ova komponenta poseduje dva važna tipa logova - *redo* i *undo* logove. Menadžer dnevnika, deo Recovery Management-a, zabeležava svaku operaciju koja se izvršava u bazi podataka. Ovaj zapis, koji se čuva kao MySQL komande, služi kao ključna evidencija za sve promene u bazi podataka. U slučaju neočekivanog pada sistema, izvršavanje ovih komandi iz logova vratilo bi bazu podataka u poslednje stabilno stanje.

Menadžer dnevnika ima ključnu odgovornost u vraćanju baze podataka u to poslednje stabilno stanje. Ovo se postiže korišćenjem dnevnika za bazu podataka, koji se dobija od menadžera bafera. Svaka operacija evidentirana u dnevniku beleži se tokom vremena od početka života baze podataka. Ova kompletna evidencija omogućava Menadžeru za oporavak da izvrši svaku komandu u datoteci evidencije, efikasno vodeći bazu podataka kroz korake oporavka.

Ovaj proces je od izuzetne važnosti jer pruža sistemsku otpornost i osigurava da baza podataka ostane konzistentna čak i nakon nepredviđenih prekida rada sistema. Menadžer za oporavak čuva integritet podataka, a primena komandi iz logova obezbeđuje da se sve promene odražavaju u bazi podataka u skladu s njihovim redosledom izvršavanja.

5.2.4. Menadžement skladištenja

Menadžment skladištenja, predstavljen na logičkom sloju MySQL arhitekture, igra ključnu ulogu u određivanju kako će podaci biti upisivani na storage. Na slici 14 se jasno uočava komponenta Storage Management, smeštena na logičkom sloju umesto na fizičkom nivou. Ova komponenta definiše strategiju upisivanja podataka na storage i često sarađuje sa menadžerom bafera kako bi efikasno upravljala memorijskim resursima.

Menadžer bafera je deo upravljanja skladištem i ima ključnu ulogu u alokaciji i dodeli memorijskih resursa. Kada izvršni mehanizam uputi zahtev, upravljanje skladištem obraća se menadžeru bafera za detalje. Ova interakcija uključuje razmenu informacija o podacima i referencama na memoriju. Menadžment skladištenja zatim prima ove podatke i referencu od menadžera bafera i vraća ih gornjem sloju arhitekture. Ovaj proces omogućava precizno upravljanje mehanizmima skladištenja, čineći ga vitalnim za efikasnost sistema. Integrisanje menadžera bafera i Upravljanja skladištem obezbeđuje optimalno korišćenje resursa i brz pristup podacima. Na taj način, Menadžment skladištenja igra ključnu ulogu u obezbeđivanju optimalne performanse i efikasnog rukovanja podacima u MySQL sistemu.

Najkorišćenije mašine za skladištenje (engl. storage engine) su InnoDB i MyISAM

InnoDB, kao jedan od najkorišćenijih storage engine-ova u MySQL arhitekturi, pruža transaction-safe mehanizam skladištenja sa sposobnostima commit-a, rollback-a i oporavka nakon pada sistema. Ovaj engine omogućava zaključavanje na nivou vrste i podržava definisanje ograničenja za strane ključeve. Sa sposobnošću da rukuje velikim tabelama, InnoDB se preporučuje za aplikacije koje zahtevaju visok integritet podataka. Klasterovani indeksi omogućavaju efikasno smanjenje I/O opterećenja, posebno za upite bazirane na primarnim ključevima.

MyISAM, kao poboljšana verzija originalnog ISAM mehanizma, predstavlja brz i efikasan storage engine. Iako nije bezbedan za transakcije, MyISAM podržava do 64 ključa po tabeli sa maksimalnom dužinom ključa od 1024 bajta. Ovaj engine se preporučuje za veb i druge aplikacije koje zahtevaju brz pristup podacima. MyISAM tabela omogućava prenos datoteka sa podacima sa sistema na sistem, ali ima ograničenja u vezi sa zaključavanjem i podrškom za strane ključeve. Osim toga, MyISAM je jedini engine koji podržava pretragu punog teksta i jednu kolonu automatskog povećanja po tabeli.

InnoDB i MyISAM, zajedno čine bitan deo MySQL koncepta, obezbeđujući različite karakteristike i performanse koje se mogu prilagoditi različitim potrebama i zahtevima aplikacija. Odluka o izboru između ova dva engine-a zavisice od specifičnih zahteva i prioriteta projekta.

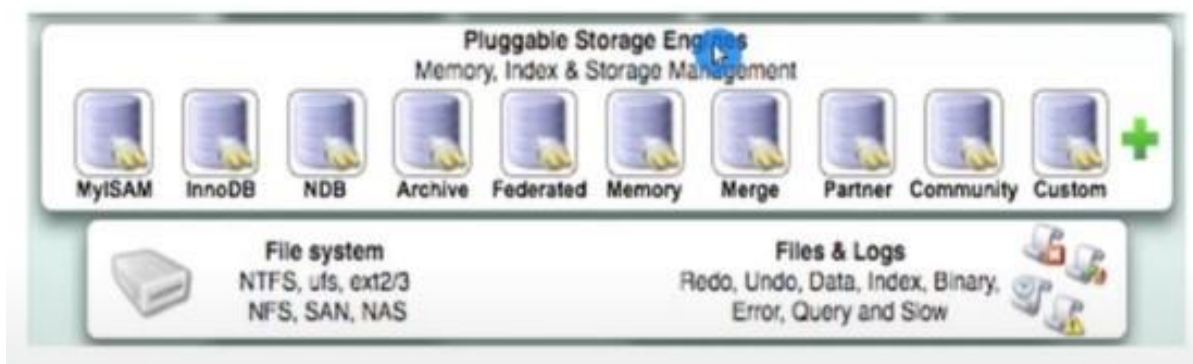
5.3. Fizički sloj

Fizički sloj MySQL arhitekture igra ključnu ulogu u upravljanju različitim vrstama informacija koje su smeštene na sekundarnom skladištu sistema. Ovaj sloj pruža neophodan okvir za efikasno čuvanje i pristup podacima, koristeći menadžere skladišta. Raznoliki tipovi podataka čine srž ovog sloja, svaki od njih sa specifičnom svrhom u kontekstu baze podataka. Tipovi podataka koji se nalaze u sistemu su sledeći:

- **Fajlovi sa podacima** (engl. data files) koji skladište korisničke podatke u bazi podataka
- **Indeksi** – koji obezbeđuju brz pristup podacima koji sadrže neke vrednosti
- **Rečnik podataka** koji skladišti metapodatke o strukturi baze podataka
- **Statistički podaci** – koji sadrže statističke informacije o podacima u bazi, koristi se od strane procesora upita za odabir efikasnog načina izvršavanja upita
- **Log informacije** – koje se koriste da prate izvršene upite tako da menadžer oporavka može da koristi te informacije da izvrši uspešan oporavak baze u slučaju pada sistema

Svi ovi tipovi podataka zajedno čine temelj fizičkog sloja MySQL arhitekture, pružajući robustan okvir za skladištenje, upravljanje i zaštitu podataka u bazi. [2]

Na gornjem delu slike 16, primetan je niz skladišnih mašina koje, kako je pomenuto u poglavlju 5.2., čine deo logičkog sloja MySQL arhitekture. Moguće ih je ukloniti ili dodati nove u skladu sa zahtevima aplikacije. Treba napomenuti da je od SQL verzije 5.1, InnoDB postala podrazumevana storage mašina, ukazujući na njenu sveprisutnost u savremenim MySQL implementacijama.



Slika 16. Mašine za skladištenje i fizički sloj [2]

Na donjem delu slike 16, prelazi se na fizički sloj arhitekture, gde se jasno vide različiti tipovi fajlova poput Redo, Undo, Data, Index, Binary, Error, i drugi. Ovi fajlovi čine suštinu fizičkog skladištenja podataka u MySQL sistemu. Svaki od njih ima specifičnu svrhu, doprinoseći integritetu, sigurnosti i efikasnosti skladištenja podataka. [2]

5.3.1. Fajlovi sa podacima

Fajlovi sa podacima (data files) predstavljaju temeljni element fizičkog sloja MySQL arhitekture, čuvajući korisničke podatke koji su ključni za funkcionisanje baze podataka. Međutim, da bi se dobio dublji uvid u organizaciju ovih fajlova, posebno se obraća pažnja na interne strukture datoteka.

Centralni koncept u organizaciji fajlova sa podacima su **stranice podataka** (engl. data pages). Stranice predstavljaju diskretne blokove podataka unutar fizičkih fajlova, obično sa fiksnom veličinom. Na primer, stranica može biti blok veličine 4KB. Svaka stranica podataka može sadržavati određeni broj redova ili zapisa iz baze podataka.

Blokovi (engl. blocks), takođe poznati kao sektori, predstavljaju osnovne jedinice podataka koje se čitaju ili pišu sa diska. Stranice podataka su organizovane unutar blokova, što omogućava efikasno upravljanje fizičkim prostorom na disku. Čitanje ili pisanje podataka obično se vrši na nivou blokova, pružajući efikasnost i optimizaciju operacija.

Unutar svake stranice podataka, podaci su organizovani u redove ili zapise. Ova organizacija omogućava efikasan pristup i manipulaciju podacima, jer se operacije često izvršavaju na nivou redova. Stranice podataka mogu sadržati i dodatne informacije, kao što su metapodaci ili kontrolne sume, koje doprinose integritetu podataka.

Dodatno, MySQL često koristi tehnike poput kompresije podataka kako bi optimizovao upotrebu prostora na disku, posebno kada je reč o velikim količinama podataka.

Sve ove interne strukture datoteka podataka pružaju robustan okvir za efikasno upravljanje i čuvanje podataka, uzimajući u obzir kako se podaci grupišu, organizuju i pristupa im se na fizičkom nivou. Ovo je ključno za obezbeđivanje optimalnih performansi i efikasnog rukovanja podacima unutar MySQL sistema

```
1  -- Kreiranje tabele
2  CREATE TABLE korisnici (
3      id INT AUTO_INCREMENT PRIMARY KEY,
4      ime VARCHAR(50),
5      prezime VARCHAR(50)
6  );
7
8  -- Unos podataka
9  INSERT INTO korisnici (ime, prezime) VALUES ('John', 'Doe');
10 INSERT INTO korisnici (ime, prezime) VALUES ('Alice', 'Smith');
```

Slika 17. Primer kreiranja tabele i dodavanja podataka

Primer sa slike 17 obuhvata kreiranje tabele korisnici sa kolonama "id," "ime," i "prezime." Nakon kreiranja tabele, koristi se INSERT INTO naredba za unos podataka u tabelu. U ovom slučaju, dodaju se dva reda sa imenima "John" i "Alice" i prezimenima "Doe" i "Smith" redom.

5.3.2. Indeksi

Indeksi u MySQL-u predstavljaju strukture podataka koje omogućavaju brz pristup podacima u tabelama baze podataka. Umesto da pretražuju celu tabelu za traženje određenih vrednosti, indeksi omogućavaju MySQL-u da efikasno locira i pristupi relevantnim podacima. Postoji nekoliko ključnih pojmova o implementaciji indeksa na fizičkom nivou u MySQL-u:

- **B-stablo** su često korišćene strukture podataka za implementaciju indeksa u MySQL-u. B-stablo su balansirana binarna stabla koja omogućavaju efikasno pretraživanje, dodavanje i brisanje elemenata. Svaki čvor B-stabla ima fiksni broj ključeva, i povezuje se sa određenim opsegom vrednosti.
- MySQL podržava dva osnovna tipa indeksa: **clustered** (klasterisani) i **non-clustered** (neklasterisani). Klasterisani indeks utiče na redosled samih redova u tabeli, dok neklasterisani indeks čuva zaseban redosled. Klasterisani indeksi često koriste B-stablo za organizaciju podataka.
- **Jedinstveni indeksi** (engl. Unique indexes) garantuju jedinstvenost vrednosti u indeksiranom polju, sprečavajući pojavu duplikata. Ovi indeksi su korisni za polja kao što su ključevi, gde svaka vrednost mora biti jedinstvena.
- **Puni tekst indeksi** (engl. Full-text indexes) omogućavaju efikasno pretraživanje teksta u poljima koja sadrže veći broj reči. Ovi indeksi koriste posebne algoritme i strukture podataka kako bi omogućili brzo pretraživanje teksta.
- **InnoDB Storage Engine**, jedan od najčešće korišćenih storage engine-ova u MySQL-u, koristi B+ stabla za implementaciju klasterisanih indeksa. Ova struktura omogućava efikasno pretraživanje podataka i podržava transakcije.
- **Analizatori i filteri** (engl. Indexing Algorithms) MySQL pruža različite analizatore i filtere koji se mogu primeniti na indekse, omogućavajući prilagodljivost zavisno o vrsti podataka. Na primer, može se koristiti puni tekst za indekse sa specifičnim analizatorima kako biste podržali pretragu teksta u različitim jezicima.

Implementacija indeksa u MySQL-u je kompleksna i zavisi od mnogo faktora, uključujući vrstu indeksa, storage engine koji se koristi, i strukturu podataka u tabelama. Kroz pametno projektovanje indeksa, moguće je značajno poboljšati performanse upita u bazi podataka.

U nastavku dati su primeri upita koji se koriste za upravljanje indeksima.

```
1 -- Kreiranje neklasterisanog indeksa na koloni ime u tabeli korisnici
2 CREATE INDEX idx_ime ON korisnici (ime);
```

Slika 18. Primer kreiranja indeksa

Primer sa slike 18 ilustruje kreiranje indeksa na tabeli korisnici. Neklasterisani indeks idx_ime se kreira na koloni "ime".

```
1 -- Prikazuje informacije o indeksima na određenoj tabeli.
2 SHOW INDEX FROM korisnici;
```

Slika 19. Upit za prikaz informacija o indeksima

	Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type
	korisnici	0	PRIMARY	1	id	A	2	NULL	NULL		BTREE
	korisnici	1	idx_ime	1	ime	A	2	NULL	NULL	YES	BTREE

Slika 20. Rezultat izvršenja upita za prikaz informacija o indeksu

Naredba sa slike 19 prikazuje informacije o indeksima na tabeli korisnici, uključujući nazive indeksa, tip indeksa, naziv kolone i ostale informacije koje se mogu videti na slici 20. Naredba sa slike 21 uklanja indeks idx_ime sa tabele korisnici.

```
1 -- Uklanjanje indeksa
2 DROP INDEX idx_ime ON korisnici;
```

Slika 21. Upit za uklanjanje indeksa

5.3.3. Rečnik podataka

Rečnik podataka u MySQL-u predstavlja ključni element u upravljanju meta podacima o strukturi baze podataka. Ovaj rečnik sadrži informacije o tome kako su tabele, kolone, indeksi, ključevi i druge strukture organizovane unutar baze podataka. Postoji nekoliko ključnih aspekata vezanih za rečnik podataka:

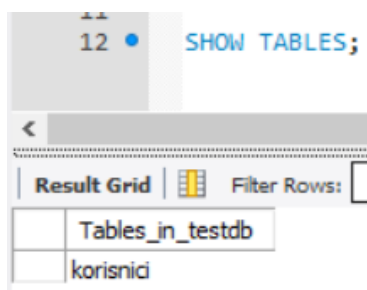
- Rečnik podataka čuva **meta podatke** o svakoj tabeli u bazi podataka. Ovo uključuje informacije o imenu tabele, broju kolona, vrstama podataka koje svaka kolona može sadržati, kao i informacije o indeksima i ključevima.
- Svaka kolona u tabeli ima svoj **opis** u rečniku podataka. Ovi opisi sadrže informacije o nazivu kolone, tipu podataka koji može sadržavati (npr. INTEGER, VARCHAR),

ograničenjima integriteta podataka (npr. NOT NULL), podrazumevanim vrednostima i drugim karakteristikama.

- Rečnik podataka sadrži **informacije o indeksima, ključevima i ograničenjima** integriteta podataka koja su definisana na tabeli. Ovi podaci pomažu MySQL-u da efikasno optimizuje upite i obezbedi integritet podataka.
- Rečnik podataka često sadrži **informacije o vezama između tabela**, uključujući strane ključeve koji povezuju jednu tabelu sa drugom. Ovo omogućava MySQL-u da efikasno izvršava upite koji uključuju JOIN operacije.
- MySQL omogućava proširivanje rečnika podataka kroz opcione karakteristike kao što su **komentari na tabelama i kolonama**, koji pružaju dodatne informacije i dokumentaciju za razvoj i održavanje baze podataka.

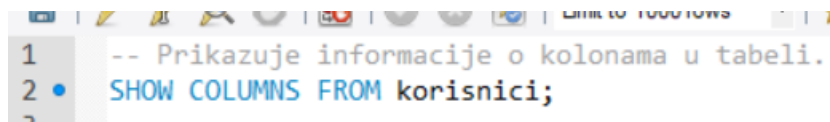
Rečnik podataka je suštinski element koji omogućava efikasno upravljanje baze podataka. Pristup ovim meta podacima pomaže MySQL-u da dinamički optimizuje izvršavanje upita, podržava integritet podataka i olakšava razvoj aplikacija koje koriste bazu podataka.

U nastavku dati su primeri MySQL naredbi koje omogućavaju rad sa rečnikom podataka. Ove naredbe omogućavaju detaljniji uvid u strukturu i karakteristike tabela u bazi podataka, pružajući korisne informacije o kolonama, indeksima, statusu tabela i drugim aspektima rečnika podataka.



Slika 22. Rezultat izvršenja upita SHOW TABLES;

Na slici 22 prikazan je rezultat izvršenja naredbe *SHOW TABLES* koja vrši prikaz svih tabela u bazi podataka, u ovom slučaju, samo tabela korisnici.



Slika 23. Upit koji prikazuje informacije o kolonama u tabeli

Na slici 23 prikazan je upit koji prikazuje informacije o kolonama u tabeli, a na slici 24 prikazan je rezultat gde se vidi da ovaj upit prikazuje informacije o svim kolonama u tabeli korisnici, uključujući tip podataka, mogućnost da li polje može biti NULL, ključeve, podrazumevane vrednosti i dodatne informacije (npr. auto_increment za id kolonu).

Result Grid						
Filter Rows:						
Export: Wrap Cell						
	Field	Type	Null	Key	Default	Extra
	id	int(11)	NO	PRI	NULL	auto increment
	ime	varchar(50)	YES	MUL	NULL	
	prezime	varchar(50)	YES		NULL	

Slika 24. Rezultat izvršenja upita SHOW columns from korisnici

```

1 -- Prikazuje informacije o svim tabelama u bazi podataka,
2 -- uključujući broj redova, veličinu, tip storage engine-a itd.
3 • SHOW TABLE STATUS LIKE 'korisnici';

```

Slika 25. Upit show table status like 'korisnici'

Upit sa slike 25 prikazuje informacije o svim tabelama u bazi podataka, uključujući broj redova, veličinu, tip storage engine-a itd., a na slici 26 prikazan je rezultat gde se može videti da tabela korisnici ima dva reda, koristi se InnoDB storage engine, kada je kreirana i slično.

Result Grid								
Filter Rows:								
Export: Wrap Cell Content: I A								
	Name	Engine	Version	Row_format	Rows	Avg_row_length	Data_length	Create_time
	korisnici	InnoDB	10	Dynamic	2	8192	16384	2024-01-21 14:39:08

Slika 26. Rezultat upita show table status like 'korisnici'

Na slici 27 prikazan upit koji se koristi za prikaz SQL naredbe koja se koristi za kreiranje određene tabele uključujući definiciju kolona, ključeva i indeksa, a na slici 28 je prikazan rezultat izvršenja upita

```

1 -- Prikazuje SQL naredbu koja se koristi za kreiranje određene tabele,
2 -- uključujući definiciju kolona, ključeva i indeksa.
3 • SHOW CREATE TABLE korisnici;

```

Slika 27. Upit show create table korisnici

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

Table	Create Table
korisnici	<pre>CREATE TABLE `korisnici` (`id` int(11) NOT NULL AUTO_INCREMENT, `ime` varchar(50) DEFAULT NULL, `prezime` varchar(50) DEFAULT NULL, PRIMARY KEY (`id`), KEY `idx_ime` (`ime`)) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8</pre>

Slika 28. Rezultat izvršenja upita show create table korisnici

5.3.4. Statistički podaci

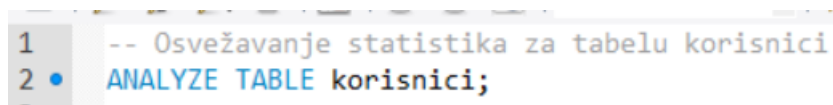
Statistički podaci u MySQL-u predstavljaju informacije o raspodeli podataka u bazi, a koriste se od strane procesora upita za optimizaciju izvršavanja upita. Ovi podaci omogućavaju MySQL-u da donosi informisane odluke o najefikasnijem načinu izvršavanja upita, poboljšavajući performanse sistema. Ključni aspekti statističkih podataka uključuju:

- Statistički podaci uključuju **histogram** koji prikazuje raspodelu vrednosti unutar određenog polja ili indeksa. Na osnovu ovih informacija, MySQL procenjuje kako su podaci raspoređeni, što pomaže u optimizaciji izbora plana izvršavanja upita.
- **Kardinalnost** se odnosi na broj jedinstvenih vrednosti u određenom polju ili indeksu. Statistički podaci pružaju procene kardinalnosti, omogućavajući MySQL-u da bolje razume veličinu skupa podataka i prilagodi strategiju izvršavanja upita.
- Kada procesor upita analizira SQL upit, statistički podaci pomažu u identifikaciji uslova selekcije (WHERE klauzule) i njihovog uticaja na ukupan broj redova u rezultatu. Ovo je ključno za optimizaciju **filtriranja podataka**.
- Na osnovu statističkih podataka, MySQL generiše plan izvršavanja upita koji određuje redosled operacija koje će se izvršavati. Ovo uključuje odabir redosleda tabela prilikom izvršavanja JOIN operacija i slično.
- MySQL redovno **osvežava statističke podatke** kako bi održavao tačnost informacija o raspodeli podataka. Ovo se obično radi automatski, ali može se ručno pokrenuti kada je potrebno.

Kroz korišćenje statističkih podataka, MySQL može prilagoditi svoje strategije izvršavanja upita, što doprinosi poboljšanju performansi i efikasnosti sistema. Ovi podaci omogućavaju MySQL-u da pravilno proceni resurse potrebne za izvršavanje upita, optimizuje korišćenje indeksa i minimizuje vreme odziva.

U nastavku dati su primeri naredbi za rad sa statističkim podacima. Ove naredbe pružaju informacije o statističkim podacima, što omogućava MySQL-u da optimizuje izvršavanje upita i poboljša performanse sistema.

Na slici 29 dat je primer naredbe koja osvežava statističke informacije za određenu tabelu, omogućavajući MySQL-u da ažurira procene raspodele podataka.



```
1 -- Osvežavanje statistika za tabelu korisnici
2 • ANALYZE TABLE korisnici;
```

Slika 29. Naredba za osvežavanje statističkih informacija

Na slici 30 dat je primer naredbe za prikaz statistike o optimizatoru upita, koja se koristi za prikazivanje statistika o različitim vrstama rukovanja podacima koje vrši MySQL server. Ove statistike često prate performanse optimizatora upita i ukazuju na način na koji server obrađuje različite tipove zahteva.

```
1 • SHOW STATUS LIKE 'Handler%';
```

Slika 30. Naredba za prikaz statistika o optimizatoru upita

Rezultat ove naredbe, prikazan na slici 31, prikazuje niz redova, gde svaki red odgovara određenom aspektu rukovanja podacima, a neki od podataka koje vraća su:

- Handler_commit - broj potvrđenih (committed) transakcija u MySQL serveru, pružajući informacije o uspešnim primenama promena u bazi podataka.
- Handler_delete statistika - broj izvršenih operacija brisanja (DELETE) nad redovima tabele u MySQL serveru.
- Handler_external_lock - broj puta kada su eksterna zaključavanja primenjena na podacima u bazi tokom operacija.
- Handler_read_first - broj puta kada je upit čitanja prvi put pročitao iz indeksa.
- Handler_read_key - Broj puta kada je ključ korišćen za čitanje reda.

Variable_name	Value
Handler commit	6
Handler delete	0
Handler discover	0
Handler external lock	30
Handler mrr init	0
Handler prepare	0
Handler read first	4
Handler read key	4
Handler read last	0
Handler read next	0
Handler read prev	0
Handler read rnd	0
Handler read rnd next	9150

Slika 31. Rezultat naredbe za prikaz statistike o optimizatoru upita

Ove statistike pružaju uvid u to koliko efikasno MySQL rukuje različitim tipovima zahteva za čitanje. Analiziranjem ovih statistika, administratori baza podataka mogu identifikovati oblasti gde može biti potrebno optimizovati indekse ili strukturu tabela kako bi poboljšali performanse sistema.

5.3.5. Log Informacije

Log informacije u MySQL-u igraju ključnu ulogu u praćenju izvršenih upita, pružajući neophodne podatke za menadžera oporavka u slučaju pada sistema. Ključne karakteristike log informacija obuhvataju:

- Log informacije beleže svaki izvršeni upit u MySQL bazi podataka, beležeći informacije o vremenu izvršavanja, korisniku koji je izvršio upit, i samom upitu.
- MySQL često koristi operativni zapis (redo log) koji sadrži promene koje treba primeniti na podacima. Ovo omogućava vraćanje sistema na tačku pre pada.
- Na osnovu log informacija, menadžer oporavka može koordinirati proces sigurnosnog kopiranja podataka. Log informacije omogućavaju precizno utvrđivanje tačke na kojoj se sistem može vratiti bez gubitka podataka. (backup)
- Log informacije prate tokove transakcija, omogućavajući MySQL-u da obezbedi konzistentnost podataka čak i nakon neočekivanih prekida sistema.
- U slučaju pada sistema ili gubitka podataka, menadžer oporavka koristi log informacije kako bi efikasno i tačno obnovio bazu podataka na poslednju konzistentnu tačku.
- Log informacije pružaju fleksibilnost u izboru tačke oporavka, omogućavajući preciznu obnovu sistema u skladu sa zahtevima.
- Kroz log informacije, MySQL održava integritet podataka, minimizujući rizik od gubitka ili nesaglasnosti podataka tokom oporavka.

Ove karakteristike log informacija omogućavaju MySQL-u da bude otporan na neočekivane događaje, obezbeđujući siguran oporavak sistema i očuvanje integriteta podataka.

U nastavku dati su primeri naredbi za rad sa log informacijama

Naredba na slici 32 prikazuje listu trenutno izvršavanih upita, uključujući informacije o vremenu izvršavanja, korisnicima i drugim detaljima. Rezultat ove naredbe prikazan je na slici 33.

```
1 -- Prikaz poslednjih izvršenih upita
2 • SHOW FULL PROCESSLIST;
```

Slika 32. Naredba show full processlist.

	Id	User	Host	db	Command	Time	State	Info
	4	root	localhost:63716	testdb	Sleep	5		NULL
	5	root	localhost:63717	testdb	Query	0	starting	SHOW FULL PROCESSLIST

Slika 33. Rezultat naredbe show full processlist

6. Zaključak

Na osnovu detaljne analize sistema za upravljanje bazama podataka (SZUBP) i posebno fokusiranja na MySQL, može se zaključiti da ova platforma igra ključnu ulogu u efikasnom upravljanju podacima u savremenim informacionim sistemima. Interna struktura MySQL skladišta podataka, koja obuhvata aplikacioni, logički i fizički sloj, otkriva ključne komponente koje čine ovu tehnologiju vitalnom za organizacije širom sveta.

Aplikacioni sloj MySQL sistema pruža korisnicima intuitivan interfejs za interakciju sa bazom podataka, omogućavajući jednostavno postavljanje upita i manipulaciju podacima. Logički sloj, sa svojim podsistemima obrade upita, upravljanja transakcijama, menadžmenta oporavka i skladištenja, predstavlja srce sistema, obezbeđujući integrisano i efikasno rukovođenje podacima.

Ipak, važno je napomenuti da, iako MySQL donosi mnoge prednosti, postoje i izazovi koji se mogu pojaviti u složenijim scenarijima ili sa velikim količinama podataka. Fizički sloj, iako efikasno upravlja skladištenjem podataka, može se suočiti sa izazovima performansi u određenim situacijama.

Uprkos potencijalnim izazovima, MySQL ostaje važan i široko korišćen alat za upravljanje bazama podataka. Razumevanje njegove interne strukture pruža osnovu za optimizaciju performansi i efikasno korišćenje u raznovrsnim poslovnim kontekstima. Ovaj rad ukazuje na značaj kontinuiranog istraživanja i unapređenja SZUBP sistema kako bi se odgovorilo na evoluirajuće zahteve modernih informacionih sistema. Literatura

7. Literatura

- [1] <https://cloudinfrastructureservices.co.uk/mysql-architecture-components-how-mysql-works-internally/>
- [2] <https://youtu.be/Szr4DrM4E8Q?list=PLd5sTGXltJ-l9PKT2Bynhg0Ou2uES0JiH>
- [3] <https://qxf2.com/blog/mysql-architecture-and-layers/>
- [4] <https://www.thegeeksearch.com/beginners-guide-to-mysql-architecture/>
- [5] <http://mysqlinternals.blogspot.com/2014/01/overview-of-mysql-architecture.html>
- [6] <https://www.rathishkumar.in/2016/04/understanding-mysql-architecture.html>
- [7] <https://chaodu.hashnode.dev/mysql-architecture>
- [8] <https://www.appdynamics.com/topics/database-management-systems>
- [9] <https://www.geeksforgeeks.org/introduction-of-3-tier-architecture-in-dbms-set-2/>
- [10] <https://www.tutorialspoint.com/Three-Level-Architecture-of-Database>