 UNIVERZITET U NIŠU

ELEKTRONSKI FAKULTET

**INTERNA STRUKTURA I KORISĆENJE INDEKSA U MYSQL BAZAMA PODATAKA**

Seminarski rad I

Studijski program: Računarstvo i informatika

Modul: Softversko inženjerstvo

Student: Mentor:

Bojan Mitić, br. indeksa: 1918 Prof. dr. Aleksandar Stanimirović

Niš, septembar 2025. Godina

**SADRŽAJ**

1. UVOD.................................................................................................................................................................................................................4

1.1 Kratak opis teme: indeksiranje u bazama podataka i njegova uloga...........................................................................................................5

1.2 Cilj rada: prikaz interne strukture indeksa u MySQL-u, korišćenje i optimizacija upita pomoću indeksa.................................................6

1.3 Motivacija: zašto je indeksiranje ključno za performanse upita u velikim bazama podataka.....................................................................9

2. ŠTA SU INDEKSI U BAZAMA PODATAKA..................................................................................................................................................10

2.1 Definicija indeksa.......................................................................................................................................................................................10

2.2 Uloga indeksa: ubrzavanje pretraga, filtriranja i sortiranja podataka.........................................................................................................12

2.3 Vrste indeksa...............................................................................................................................................................................................13

2.4 Prednosti i mane korišćenja indeksa...........................................................................................................................................................17

3. INTERNA STRUKTURA INDEKSA U MYSQL-U………………………………………………………………………………………….18

3.1 Tipovi skladištenja (Storage Engines) u MySQL-u………………………………………………………………………………………18

3.2 B-Tree: Primarni i sekundarni indeksi……………………………………………………………………………………………………19

3.3 Clustered vs Non-clustered indeksi………………………………………………………………………………………………………19

3.4 Interna organizacija: listovi i čvorovi B-tree structure…………………………………………………………………………………...19

3.5 MyISAM: B-Tree indeksiranje i odvojena struktura za podatke i indeks………………………………………………………………..21

3.6 B-Tree indeksiranje u MyISAM-u………………………………………………………………………………………………………..22

3.7 Odvojena struktura za podatke i indeks…………………………………………………………………………………………………..22

3.8 Prednosti i ograničenja…………………………………………………………………………………………………………………...22

3.9 Kako MySQL održava statistiku indeksa………………………………………………………………………………………………...23

3.10 Fragmentacija i ažuriranje indeksa……………………………………………………………………………………………………...26

4. KORIŠĆENJE INDEKSA U UPITIMA........................................................................................................................................................... 28

4.1 Načini na koje MySQL koristi indekse......................................................................................................................................................28

4.2 WHERE uslovi...........................................................................................................................................................................................29

4.3 JOIN operacije...........................................................................................................................................................................................29

4.4 ORDER BY i GROUP BY.........................................................................................................................................................................30

4.5 LIKE sa prefiksom.....................................................................................................................................................................................30

5. OPTIMIZACIJA UPITA POMOĆU INDEKSA...............................................................................................................................................32

5.1 Korišćenje EXPLAIN za analizu upita......................................................................................................................................................33

5.2 Composite indeksi i redosled kolona.........................................................................................................................................................34

5.3 Preporuke za smanjenje nepotrebnog indeksiranja....................................................................................................................................35

6. PRAKTIČAN DEO-DEMONSTRACIJA U MYSQL-U……………………………………………………………………………………..37

6.1 Kreiranje baze i tabela……………………………………………………………………………………………………………………37

6.2 Popunjavanje tabela podacima……………………………………………………………………………………………………………38

6.3 Korišćenje indeksa u upitima……………………………………………………………………………………………………………..39

6.4 Analiza upita pomoću EXPLAIN…………………………………………………………………………………………………………40

6.5 Rezultati testiranja na MySQL client-u…………………………………………………………………………………………………...41

7. ZAKLJUČAK......................................................................................................................................................................................................41

7.1 Preporuke za efikasno indeksiranje u MySQL-u........................................................................................................................................42

7.2 Smernice za dalji razvoj..............................................................................................................................................................................42

8. REFERENCE......................................................................................................................................................................................................43

1. **UVOD**

U savremenom digitalnom dobu, upravljanje podacima predstavlja temeljnu komponentu gotovo svakog informacionog sistema. Baze podataka su postale neizostavan alat u organizaciji, skladištenju i pretraživanju velikih količina informacija, a među najzastupljenijim sistemima za upravljanje bazama podataka (DBMS) ističe se MySQL — otvoreni, robusni i skalabilni sistem koji se koristi širom sveta, od malih aplikacija do kompleksnih enterprise rešenja.

Efikasnost rada sa bazama podataka ne zavisi samo od pravilnog modelovanja podataka, već i od optimizacije pristupa tim podacima. U tom kontekstu, indeksi igraju ključnu ulogu. Oni omogućavaju brže izvršavanje upita, smanjujući potrebu za sekvencijalnim pretraživanjem velikih tabela. Bez indeksa, čak i najjednostavniji upiti mogu postati izuzetno spori kada se primenjuju na tabele sa velikim brojem redova. Upravo zbog toga, razumevanje interne strukture indeksa u MySQL-u, kao i njihovo pravilno korišćenje, predstavlja važan korak ka izgradnji efikasnih i skalabilnih aplikacija.

Ovaj seminarski rad ima za cilj da detaljno istraži unutrašnju organizaciju indeksa u MySQL bazama podataka, objasni različite tipove indeksa koji su dostupni korisnicima, kao i da pruži uvid u načine na koje se indeksi kreiraju, održavaju i koriste u realnim scenarijima. Poseban fokus biće stavljen na B-Tree i Hash indekse, koji se najčešće koriste u okviru MySQL-ovih skladišnih mehanizama kao što su InnoDB i Memory. Takođe, biće razmotreni i koncepti primarnih i stranih ključeva, jedinstvenih indeksa, kao i složenih (kompozitnih) indeksa, uz analizu njihovog uticaja na performanse upita.

Pored teorijskog okvira, rad će obuhvatiti i praktične primere implementacije indeksa, demonstraciju njihovog uticaja na brzinu izvršavanja SQL upita, kao i preporuke za optimalno korišćenje indeksa u različitim situacijama — od jednostavnih CRUD operacija do kompleksnih analitičkih upita. Na kraju, biće razmotreni i potencijalni izazovi i greške koje se mogu javiti pri radu sa indeksima, kao što su prekomerna indeksacija, neefikasno korišćenje složenih indeksa ili zanemarivanje statistike baze.

Razumevanje interne strukture i pravilno korišćenje indeksa u MySQL-u ne samo da doprinosi boljoj performansi aplikacija, već i omogućava programerima, administratorima i analitičarima da donose informisane odluke prilikom dizajniranja i optimizacije baza podataka. U tom smislu, ovaj rad predstavlja korak ka dubljem razumevanju jednog od najvažnijih aspekata rada sa relacionim bazama podataka.

* 1. **Kratak opis teme: indeksiranje u bazama podataka i njegova uloga**

U eri digitalne transformacije, baze podataka predstavljaju temeljnu infrastrukturu za skladištenje, organizaciju i pristup informacijama. Bez obzira da li se radi o malim lokalnim aplikacijama ili kompleksnim distribuiranim sistemima, efikasnost rada sa podacima direktno utiče na performanse celokupnog sistema. Jedan od ključnih mehanizama koji omogućava brzo i efikasno pretraživanje podataka jeste **indeksiranje**.

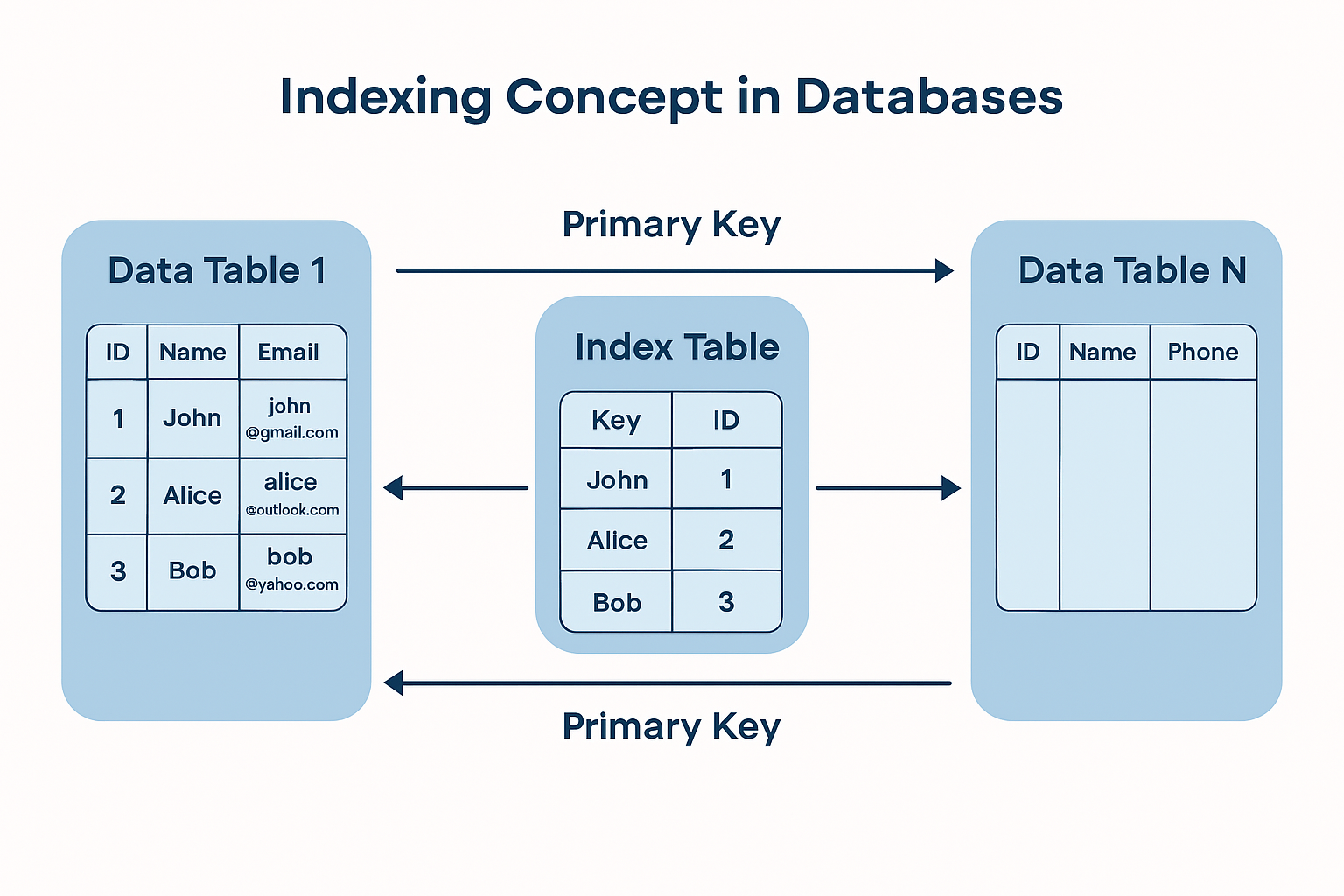
Indeksiranje u bazama podataka funkcioniše po analogiji sa indeksima u knjigama — umesto da se svaki put pretražuje cela tabela, indeks omogućava direktan pristup relevantnim podacima, značajno smanjujući vreme izvršavanja upita. U tehničkom smislu, indeks je posebna struktura podataka koja se kreira nad jednom ili više kolona tabele, sa ciljem da optimizuje operacije pretrage, sortiranja i filtriranja. Najčešće korišćeni tipovi indeksa u MySQL bazama podataka su **B-Tree** i **Hash** indeksi, pri čemu svaki ima specifične prednosti u zavisnosti od tipa upita i skladišnog mehanizma (npr. InnoDB, MyISAM, Memory).

Uloga indeksa nije ograničena samo na ubrzavanje SELECT upita. Oni imaju ključnu funkciju i u implementaciji **primarnih ključeva**, **jedinstvenih ograničenja**, kao i u održavanju **referencijalnog integriteta** kroz strane ključeve. Pravilno dizajnirani indeksi mogu značajno poboljšati performanse aplikacije, dok loše postavljeni ili prekomerni indeksi mogu dovesti do usporenja prilikom INSERT, UPDATE i DELETE operacija, kao i do povećanja potrošnje memorije.

U okviru ovog seminarskog rada biće detaljno razmotreni:

* osnovni principi indeksiranja,
* vrste indeksa i njihove interne strukture,
* način kreiranja i održavanja indeksa u MySQL-u,
* uticaj indeksa na performanse baze podataka,
* praktični primeri optimizacije upita korišćenjem indeksa.

Razumevanje indeksiranja predstavlja neophodan korak ka profesionalnom radu sa bazama podataka, jer omogućava ne samo tehničku efikasnost, već i strateško planiranje arhitekture sistema. U nastavku je prikazana ilustracija koja vizualizuje osnovni koncept indeksiranja u relacionim bazama podataka:



* 1. **Cilj rada: prikaz interne strukture indeksa u MySQL-u, korišćenje i optimizacija upita pomoću indeksa**

U savremenim informacionim sistemima, brzina i efikasnost pristupa podacima predstavljaju ključne faktore uspešnosti aplikacija koje se oslanjaju na baze podataka. Kako se obim podataka eksponencijalno povećava, potreba za optimizacijom upita postaje sve izraženija. Jedan od najvažnijih mehanizama koji omogućava tu optimizaciju jeste indeksiranje — proces kreiranja posebnih struktura podataka koje omogućavaju brži pristup informacijama u tabelama. U okviru MySQL sistema za upravljanje bazama podataka, indeksi igraju centralnu ulogu u poboljšanju performansi, održavanju integriteta podataka i efikasnom izvršavanju upita.

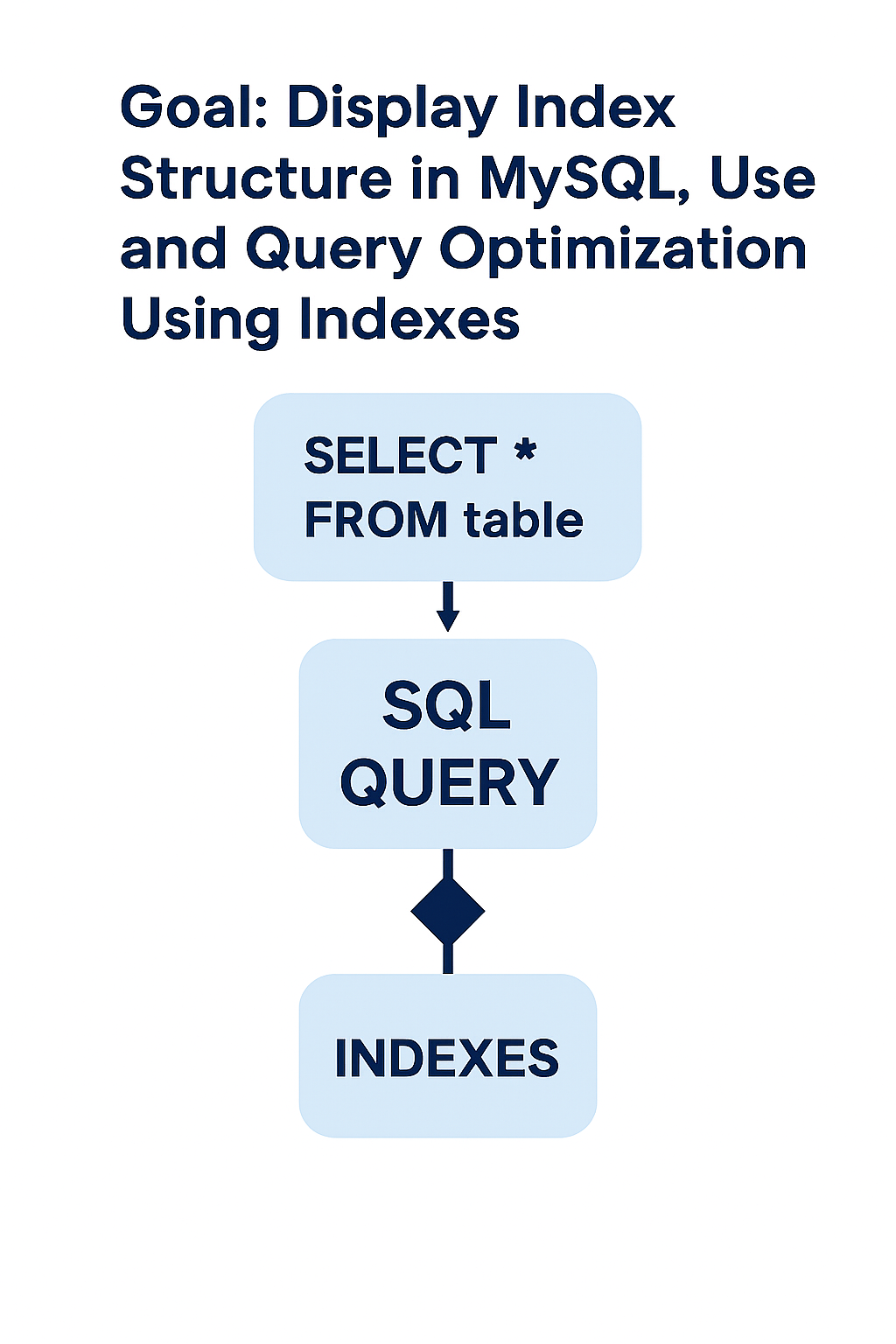
Cilj ovog seminarskog rada jeste da pruži sveobuhvatan pregled interne strukture indeksa u MySQL-u, sa posebnim osvrtom na način na koji se indeksi implementiraju, održavaju i koriste u realnim aplikacijama. Rad će obuhvatiti teorijski okvir koji objašnjava osnovne principe indeksiranja, kao i tehničke detalje o vrstama indeksa koje MySQL podržava — uključujući B-Tree, Hash, Fulltext i Spatial indekse. Biće analizirana razlika između primarnih, jedinstvenih, stranih i složenih (kompozitnih) indeksa, uz objašnjenje njihove funkcionalnosti i uticaja na performanse baze.

Poseban fokus biće stavljen na **internu strukturu indeksa**, odnosno način na koji se podaci organizuju unutar indeksnih stabala, kako se održava ravnoteža u B-Tree strukturama, i kako se koristi binarna pretraga za pronalaženje vrednosti. Razumevanje ovih mehanizama omogućava dublji uvid u to kako MySQL optimizuje izvršavanje upita, naročito kada se radi o velikim tabelama sa milionima redova.

Pored teorijskog aspekta, rad će sadržati i **praktične primere** kreiranja i korišćenja indeksa u MySQL-u, uključujući demonstraciju uticaja indeksa na brzinu izvršavanja SELECT, JOIN i WHERE upita. Biće prikazani konkretni slučajevi u kojima se korišćenjem indeksa vreme izvršavanja upita smanjuje sa nekoliko sekundi na milisekunde, čime se jasno ilustruje njihova vrednost u optimizaciji performansi.

Takođe, rad će obuhvatiti **analizu potencijalnih izazova** u radu sa indeksima, kao što su prekomerna indeksacija, neefikasno korišćenje složenih indeksa, kao i problemi sa ažuriranjem podataka u tabelama sa velikim brojem indeksa. Biće predstavljene preporuke za optimalno indeksiranje, uključujući strategije za balansiranje između brzine čitanja i brzine pisanja podataka.

Konačni cilj rada jeste da studentima, programerima i administratorima baza podataka pruži jasno, strukturirano i primenljivo znanje o indeksima u MySQL-u, koje mogu koristiti za dizajniranje efikasnih baza, optimizaciju upita i rešavanje problema sa performansama. Razumevanje interne strukture indeksa i njihovog uticaja na rad baze predstavlja temelj za profesionalni rad u oblasti upravljanja podacima i razvoja aplikacija koje se oslanjaju na relacioni model.

****

* 1. **Motivacija: zašto je indeksiranje ključno za performanse upita u velikim bazama podataka**

U savremenim informacionim sistemima, količina podataka koja se skladišti i obrađuje raste iz dana u dan. Organizacije, aplikacije i servisi generišu ogromne količine informacija koje se čuvaju u bazama podataka — od korisničkih profila i transakcija, do logova sistema i analitičkih izveštaja. U takvom okruženju, brzina pristupa podacima postaje kritičan faktor uspešnosti sistema. Bez adekvatnih mehanizama za optimizaciju, čak i najjednostavniji upiti mogu postati neefikasni, dovodeći do usporenja aplikacija, povećanja latencije i nezadovoljstva korisnika.

Jedan od najefikasnijih i najrasprostranjenijih mehanizama za poboljšanje performansi upita jeste **indeksiranje**. Indeksi omogućavaju brži pristup podacima tako što eliminišu potrebu za sekvencijalnim pretraživanjem cele tabele. Umesto da baza pregleda svaki red kako bi pronašla odgovarajuću vrednost, indeks omogućava direktan pristup relevantnim redovima pomoću strukturirane pretrage — najčešće kroz B-Tree strukture koje omogućavaju logaritamsku složenost pretrage.

Motivacija za detaljno razmatranje indeksiranja u ovom radu proizlazi iz sledećih ključnih razloga:

* **Eksponencijalni rast podataka**: U velikim bazama koje sadrže milione ili čak milijarde redova, neoptimizovani upiti mogu trajati više sekundi ili minuta, što je neprihvatljivo u realnom vremenu. Indeksi omogućavaju da se vreme izvršavanja upita smanji na milisekunde.
* **Povećanje konkurentnosti sistema**: U višekorisničkim okruženjima, gde se istovremeno izvršava veliki broj upita, indeksiranje smanjuje opterećenje na sistem i omogućava efikasniju raspodelu resursa.
* **Optimizacija JOIN operacija**: U relacionim bazama podataka, spajanje tabela je česta operacija. Indeksi na kolone koje se koriste za povezivanje omogućavaju da se JOIN izvrši brzo i efikasno, bez potrebe za potpunim skeniranjem tabela.
* **Podrška za filtriranje i sortiranje**: WHERE, ORDER BY i GROUP BY klauzule u SQL-u mogu biti izuzetno zahtevne bez indeksa. Pravilno postavljeni indeksi omogućavaju da se ove operacije izvrše uz minimalnu potrošnju resursa.
* **Održavanje integriteta podataka**: Indeksi se koriste i za implementaciju primarnih i stranih ključeva, čime se obezbeđuje konzistentnost i validnost podataka u tabelama.
* **Balans između brzine čitanja i pisanja**: Iako indeksi mogu usporiti operacije pisanja (INSERT, UPDATE, DELETE), pravilno dizajnirana strategija indeksiranja omogućava optimalan balans između performansi čitanja i ažuriranja podataka.

U kontekstu MySQL-a, koji se često koristi u web aplikacijama, analitičkim sistemima i servisima sa velikim brojem korisnika, indeksiranje predstavlja neophodan alat za skalabilnost i pouzdanost. Razumevanje interne strukture indeksa, njihovog uticaja na izvršavanje upita i pravilno korišćenje u dizajnu baze podataka, omogućava programerima i administratorima da izgrade sisteme koji su ne samo funkcionalni, već i izuzetno efikasni.

Zbog svega navedenog, motivacija za ovaj rad leži u potrebi da se indeksiranje ne posmatra kao tehnička formalnost, već kao strateški alat za optimizaciju performansi, skalabilnost sistema i kvalitet korisničkog iskustva.

1. **ŠTA SU INDEKSI U BAZAMA PODATAKA**

* 1. **Definicija indeksa**

Indeksi u bazama podataka predstavljaju posebne strukture podataka koje omogućavaju brži i efikasniji pristup informacijama u tabelama. Njihova osnovna funkcija je da ubrzaju izvršavanje upita tako što omogućavaju direktno pronalaženje redova koji zadovoljavaju određene kriterijume, bez potrebe za sekvencijalnim pretraživanjem cele tabele. U tom smislu, indeks se može posmatrati kao mapa koja vodi ka traženim podacima, slično kao što indeks na kraju knjige omogućava brzo pronalaženje stranice na kojoj se nalazi određeni pojam.

Tehnički gledano, indeks je struktura koja se kreira nad jednom ili više kolona tabele, i koja sadrži organizovane vrednosti tih kolona zajedno sa referencama (najčešće primarnim ključevima) ka redovima u tabeli. Kada se izvrši upit koji koristi kolonu nad kojom postoji indeks, baza koristi indeksnu strukturu da brzo pronađe odgovarajuće redove, umesto da pregleda svaki red pojedinačno.

U MySQL-u, najčešće korišćeni tipovi indeksa su:

* **B-Tree indeks**: koristi se u većini skladišnih mehanizama (npr. InnoDB, MyISAM) i omogućava efikasnu pretragu, sortiranje i opsežne upite (npr. WHERE x BETWEEN a AND b).
* **Hash indeks**: koristi se u Memory skladišnom mehanizmu i omogućava brzu tačnu pretragu, ali nije pogodan za opsege ili sortiranje.
* **Fulltext indeks**: koristi se za pretragu teksta u velikim tekstualnim poljima, omogućavajući pretragu po ključnim rečima.
* **Spatial indeks**: koristi se za rad sa geolokacijskim podacima i prostornim objektima.

Indeksi se dele i prema funkcionalnosti:

* **Primarni indeks (PRIMARY KEY)**: automatski se kreira nad kolonom koja je definisana kao primarni ključ. Obezbeđuje jedinstvenost i koristi se za identifikaciju redova.
* **Jedinstveni indeks (UNIQUE)**: obezbeđuje da vrednosti u indeksiranoj koloni budu jedinstvene.
* **Obični (neunikatni) indeks**: koristi se za ubrzavanje pretrage, ali ne garantuje jedinstvenost vrednosti.
* **Složeni (kompozitni) indeks**: kreira se nad više kolona i koristi se kada se upiti oslanjaju na kombinaciju vrednosti iz tih kolona.

Indeksi imaju ključnu ulogu u optimizaciji performansi baze podataka. Bez indeksa, baza mora da izvrši tzv. **full table scan**, što znači da mora da pregleda svaki red u tabeli da bi pronašla odgovarajuće podatke. To može biti izuzetno sporo, naročito kod velikih tabela sa stotinama hiljada ili milionima redova. Sa indeksima, baza koristi strukturirani pristup koji omogućava da se relevantni podaci pronađu brzo i efikasno.

Međutim, indeksiranje nije bez kompromisa. Svaki indeks zauzima dodatni prostor na disku i u memoriji, a operacije kao što su INSERT, UPDATE i DELETE mogu biti sporije jer se indeksne strukture moraju ažurirati. Zbog toga je važno pažljivo planirati koje kolone treba indeksirati, u skladu sa tipičnim upitima koji se izvršavaju nad bazom.

U zaključku, indeksi predstavljaju jedan od najvažnijih alata za optimizaciju rada sa bazama podataka. Njihovo pravilno korišćenje omogućava značajno poboljšanje performansi, smanjenje vremena izvršavanja upita i efikasnije korišćenje resursa. Razumevanje strukture, tipova i funkcionalnosti indeksa je neophodno za svakog ko se bavi dizajnom, administracijom ili optimizacijom baza podataka.

* 1. **Uloga indeksa: ubrzavanje pretraga, filtriranja i sortiranja podataka**

U radu sa bazama podataka, brzina pristupa informacijama predstavlja jedan od najvažnijih faktora koji direktno utiče na performanse aplikacija, korisničko iskustvo i efikasnost sistema. Kako se obim podataka u tabelama povećava, potreba za optimizacijom upita postaje sve izraženija. Upravo tu indeksiranje igra ključnu ulogu — kao mehanizam koji omogućava da se podaci pronađu, filtriraju i sortiraju na efikasan način, bez potrebe za pregledom svakog reda u tabeli.

**1. Ubrzavanje pretrage podataka**

Bez indeksa, baza podataka mora da izvrši tzv. *full table scan*, odnosno da pregleda svaki red u tabeli kako bi pronašla podatke koji zadovoljavaju uslove upita. Ova operacija je izuzetno neefikasna kod velikih tabela. Kada postoji indeks nad relevantnom kolonom, baza koristi strukturirani pristup (npr. B-Tree) da brzo pronađe tražene vrednosti. Na taj način, vreme izvršavanja upita se smanjuje sa linearne složenosti (O(n)) na logaritamsku (O(log n)), što predstavlja ogromnu razliku kod velikih datasetova.

**2. Efikasno filtriranje podataka**

Indeksi omogućavaju da se WHERE klauzule u SQL upitima izvršavaju znatno brže. Na primer, upit poput SELECT \* FROM korisnici WHERE grad = 'Beograd' će se izvršiti daleko efikasnije ako postoji indeks nad kolonom „grad“. Umesto da se svaki red proverava pojedinačno, baza koristi indeks da pronađe sve redove koji odgovaraju uslovu, čime se značajno smanjuje broj operacija čitanja.

**3. Optimizacija sortiranja**

ORDER BY klauzula u SQL-u često zahteva dodatne resurse, naročito kada se radi nad velikim skupovima podataka. Ako postoji indeks nad kolonom po kojoj se vrši sortiranje, baza može da koristi već sortirane vrednosti iz indeksne strukture, čime se izbegava dodatno sortiranje u memoriji ili na disku. Ovo je posebno korisno kod paginacije rezultata (LIMIT, OFFSET) i kod prikaza podataka u korisničkim interfejsima.

**4. Kombinacija sa JOIN operacijama**

U relacionim bazama podataka, spajanje tabela (JOIN) je česta operacija. Kada se JOIN vrši nad kolonama koje su indeksirane, baza može da pronađe povezane redove brzo i efikasno, bez potrebe za kombinovanjem svakog reda iz jedne tabele sa svakim redom iz druge. Ovo značajno smanjuje kompleksnost operacije i ubrzava izvršavanje upita.

**5. Podrška za agregaciju i grupisanje**

Indeksi mogu pomoći i kod GROUP BY i HAVING klauzula, jer omogućavaju da se podaci grupišu na osnovu već organizovanih vrednosti. Ovo je posebno korisno u analitičkim upitima gde se vrši agregacija nad velikim brojem redova.

U zaključku, indeksi nisu samo tehnički dodatak tabelama — oni su strateški alat za optimizaciju performansi. Njihova pravilna primena omogućava da se upiti izvršavaju brže, da se resursi sistema koriste efikasnije i da se korisnicima obezbedi brz i pouzdan pristup informacijama. U kontekstu velikih baza podataka, indeksiranje je neophodno za skalabilnost i stabilnost sistema.

* 1. **Vrste indeksa**

Indeksi u MySQL bazama podataka dolaze u više oblika, od kojih svaki ima specifičnu namenu i primenu u zavisnosti od tipa podataka, strukture tabele i zahteva aplikacije. Pravilno razumevanje i korišćenje različitih tipova indeksa omogućava značajno poboljšanje performansi upita, održavanje integriteta podataka i efikasno upravljanje velikim količinama informacija. U nastavku su opisane najvažnije vrste indeksa koje MySQL podržava:

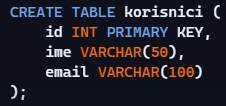
**1. Primarni indeks (PRIMARY KEY)**

Primarni indeks se automatski kreira kada se u tabeli definiše primarni ključ. On obezbeđuje jedinstvenost vrednosti u koloni (ili kombinaciji kolona) i koristi se za jednoznačnu identifikaciju svakog reda u tabeli. U skladišnom mehanizmu InnoDB, primarni indeks se implementira kao klasterovani indeks, što znači da se podaci u tabeli fizički organizuju prema vrednostima primarnog ključa.

**Karakteristike:**

* Ne dozvoljava duplikate.
* Ne dozvoljava NULL vrednosti.
* Automatski se koristi za povezivanje tabela putem stranih ključeva.

**Primer:**



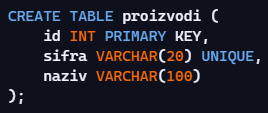
**2. Jedinstveni indeks (UNIQUE)**

Jedinstveni indeks obezbeđuje da vrednosti u indeksiranoj koloni budu jedinstvene, ali za razliku od primarnog indeksa, dozvoljava NULL vrednosti (osim ako nije drugačije definisano). Koristi se kada je potrebno sprečiti duplikate u koloni, ali bez potrebe da ta kolona bude primarni ključ.

**Karakteristike:**

* Sprečava duplikate.
* Može se kreirati nad više kolona (kompozitni indeks).
* Dozvoljava više NULL vrednosti u nekim skladišnim mehanizmima.

**Primer:**

****

**3. Obični indeks (INDEX)**

Obični indeks, poznat i kao neunikatni indeks, koristi se isključivo za poboljšanje performansi upita. Ne nameće ograničenja na jedinstvenost vrednosti i može se kreirati nad jednom ili više kolona. Najčešće se koristi za optimizaciju WHERE, ORDER BY i JOIN operacija.

**Karakteristike:**

* Ne garantuje jedinstvenost.
* Može se kreirati nad bilo kojom kolonom.
* Koristi se za ubrzavanje pretrage i sortiranja.

**Primer:**

****

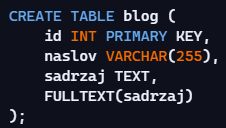
**4. Fulltext indeks (FULLTEXT)**

Fulltext indeks se koristi za pretragu velikih tekstualnih sadržaja, kao što su članci, opisi proizvoda ili komentari. Omogućava pretragu po ključnim rečima, frazama i podržava napredne funkcije kao što su rangiranje rezultata i logički operatori (AND, OR).

**Karakteristike:**

* Dostupan u InnoDB i MyISAM mehanizmima.
* Radi samo nad tekstualnim kolonama (CHAR, VARCHAR, TEXT).
* Koristi se uz MATCH ... AGAINST sintaksu.

**Primer:**

****

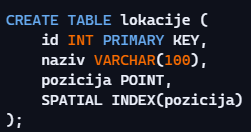
**5. Prostorni indeks (SPATIAL)**

Prostorni indeks se koristi za rad sa geolokacijskim i prostornim podacima, kao što su tačke, linije i poligoni. Omogućava efikasnu pretragu i analizu prostornih odnosa, kao što su udaljenost, preklapanje i sadržavanje.

**Karakteristike:**

* Dostupan u MySQL-u uz SPATIAL tipove podataka (POINT, LINESTRING, POLYGON).
* Koristi se u aplikacijama koje rade sa mapama, GPS podacima i geoinformacionim sistemima.
* Zahteva da kolona bude definisana kao prostorni tip.

**Primer:**

****

U zaključku, izbor odgovarajućeg tipa indeksa zavisi od konkretnih potreba baze podataka i tipičnih upita koji se nad njom izvršavaju. Kombinovanjem različitih indeksa moguće je postići optimalan balans između brzine pristupa podacima, integriteta baze i efikasnosti izvršavanja operacija. U narednim poglavljima biće prikazani praktični primeri korišćenja ovih indeksa i njihov uticaj na performanse.



* 1. **Prednosti i mane korišćenja indeksa**

Indeksi predstavljaju jedan od najvažnijih alata za optimizaciju rada sa bazama podataka. Njihova pravilna primena može značajno poboljšati performanse sistema, dok neadekvatno korišćenje može dovesti do suprotnog efekta — usporenja, povećanja potrošnje resursa i komplikacija u održavanju baze. U ovom poglavlju razmatraju se ključne prednosti i potencijalne mane korišćenja indeksa u MySQL bazama podataka.

**✅ Prednosti korišćenja indeksa**

* **Ubrzanje izvršavanja upita** Indeksi omogućavaju brzu pretragu podataka, čime se vreme izvršavanja SELECT, JOIN, WHERE, ORDER BY i GROUP BY upita značajno smanjuje. Umesto da baza pregleda svaki red u tabeli, koristi indeksnu strukturu za direktan pristup relevantnim redovima.
* **Efikasno sortiranje i filtriranje** Kada se koristi ORDER BY ili WHERE nad indeksiranom kolonom, baza koristi već organizovane podatke iz indeksa, čime se izbegava dodatno sortiranje u memoriji.
* **Poboljšanje performansi kod spajanja tabela (JOIN)** Indeksi nad kolonama koje se koriste za povezivanje tabela omogućavaju da se JOIN operacije izvršavaju brzo i efikasno, bez potrebe za potpunim skeniranjem tabela.
* **Održavanje integriteta podataka** Indeksi se koriste za implementaciju primarnih i stranih ključeva, čime se obezbeđuje konzistentnost i validnost podataka u tabelama.
* **Pomoć kod agregacije i grupisanja** Kod GROUP BY i HAVING klauzula, indeksirane kolone omogućavaju brže grupisanje i agregaciju podataka.
* **Podrška za fulltext pretragu i prostorne podatke** Specijalizovani indeksi kao što su FULLTEXT i SPATIAL omogućavaju naprednu pretragu teksta i rad sa geolokacijskim informacijama.

**⚠️ Mane korišćenja indeksa**

* **Povećana potrošnja memorije i prostora na disku** Svaki indeks zauzima dodatni prostor, što može biti značajno kod velikih tabela sa više indeksiranih kolona.
* **Usporavanje operacija pisanja (INSERT, UPDATE, DELETE)** Prilikom izmene podataka, indeksne strukture moraju da se ažuriraju, što može usporiti operacije pisanja, naročito kod tabela sa velikim brojem indeksa.
* **Prekomerna indeksacija** Dodavanje indeksa bez jasne potrebe može dovesti do komplikacija u održavanju baze, povećanja kompleksnosti i smanjenja ukupne efikasnosti.
* **Neefikasnost kod malih tabela** Kod tabela sa malim brojem redova, indeksiranje često ne donosi značajne performanse, a može čak i usporiti rad zbog dodatne logike koju baza mora da obradi.
* **Pogrešan izbor kolona za indeksiranje** Ako se indeksira kolona koja se retko koristi u upitima, indeks neće imati efekta. Takođe, indeksiranje kolona sa velikim brojem ponavljajućih vrednosti (npr. pol „muški/ženski“) često nije korisno.

U zaključku, indeksi su moćan alat za optimizaciju rada sa bazama podataka, ali zahtevaju pažljivo planiranje i analizu. Pravilno korišćenje indeksa donosi značajne performanse, dok nepromišljeno indeksiranje može dovesti do problema u skalabilnosti i održavanju sistema. Zbog toga je važno da se indeksiranje vrši u skladu sa konkretnim potrebama aplikacije, tipičnim upitima i karakteristikama podataka.

1. **INTERNA STRUKTURA INDEKSA U MYSQL-U**

Efikasno indeksiranje predstavlja ključni aspekt performansi u relacionim bazama podataka. U MySQL-u, indeksne strukture se implementiraju kroz različite skladišne mehanizme (Storage Engines), od kojih je najzastupljeniji i najmoćniji **InnoDB**. Razumevanje interne organizacije indeksa u okviru InnoDB-a omogućava dublji uvid u način na koji se podaci pretražuju, sortiraju i povezuju, kao i kako se optimizuju SQL upiti.

* 1. **Tipovi skladištenja (Storage Engines) u MySQL-u**

MySQL podržava više skladišnih mehanizama, uključujući MyISAM, Memory, CSV, Archive, ali **InnoDB** je podrazumevani engine u modernim verzijama. On nudi napredne funkcionalnosti kao što su:

* Transakcije sa podrškom za ACID principe
* Zaključavanje na nivou reda
* Automatsko obnavljanje nakon pada sistema
* Podrška za strane ključeve i referencijalni integritet

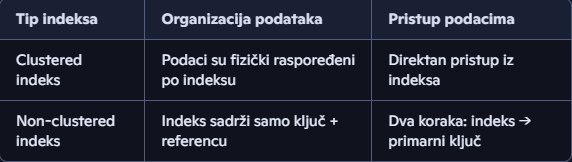
Najvažnija karakteristika InnoDB-a u kontekstu indeksiranja jeste korišćenje **B-Tree struktura** za organizaciju podataka i indeksa.

* 1. **B-Tree: Primarni i sekundarni indeksi**

U InnoDB-u, svi indeksi se baziraju na **B-Tree** strukturi, koja omogućava logaritamsku složenost pretrage, umetanja i brisanja. Postoje dve osnovne vrste indeksa:

* **Primarni indeks (PRIMARY KEY)** U InnoDB-u, primarni indeks je ujedno i **clustered indeks**, što znači da se podaci u tabeli fizički organizuju prema vrednostima primarnog ključa. Svaki list B-Tree stabla sadrži ceo red tabele.
* **Sekundarni indeks (INDEX, UNIQUE)** Sekundarni indeksi su **non-clustered**. Oni sadrže vrednosti indeksiranih kolona i internu referencu ka primarnom ključu. Prilikom pretrage, baza koristi sekundarni indeks da pronađe primarni ključ, a zatim koristi primarni indeks da dođe do celog reda.
  1. **Clustered vs Non-clustered indeksi**

Razlika između ove dve vrste indeksa je ključna za razumevanje performansi:

****

**Clustered indeks** omogućava brži pristup jer se podaci nalaze direktno u listovima stabla. Kod **non-clustered indeksa**, pristup zahteva dodatnu operaciju — prvo se pronalazi vrednost u sekundarnom indeksu, a zatim se koristi primarni ključ da bi se dohvatio ceo red.

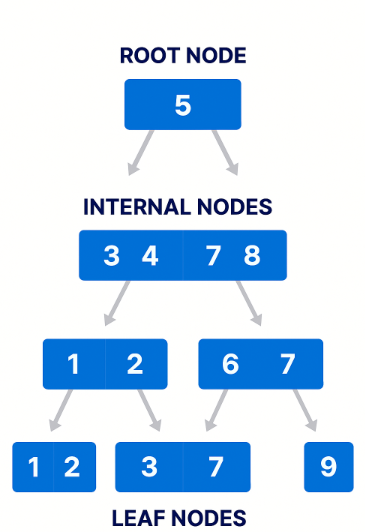
* 1. **Interna organizacija: listovi i čvorovi B-Tree strukture**

B-Tree struktura u InnoDB-u organizovana je hijerarhijski:

* **Korenski čvor (Root Node):** Početna tačka pretrage. Sadrži pokazivače ka unutrašnjim čvorovima i ključne vrednosti koje određuju opseg.
* **Unutrašnji čvorovi (Internal Nodes):** Služe za navigaciju kroz stablo. Ne sadrže podatke iz tabele, već samo ključeve i pokazivače ka nižim čvorovima.
* **Listovi (Leaf Nodes):** Najniži nivo stabla. Kod primarnog indeksa, listovi sadrže ceo red tabele. Kod sekundarnih indeksa, listovi sadrže vrednosti indeksiranih kolona i referencu ka primarnom ključu.

Ova organizacija omogućava da se podaci pretražuju efikasno, bez potrebe za sekvencijalnim pregledom cele tabele. B-Tree struktura se automatski balansira prilikom umetanja i brisanja, čime se održava optimalna dubina stabla i konstantne performanse.

U zaključku, interna struktura indeksa u MySQL-u, posebno u okviru InnoDB skladišnog mehanizma, predstavlja sofisticiran sistem koji omogućava visoku efikasnost u radu sa velikim količinama podataka. Razumevanje razlike između primarnih i sekundarnih indeksa, kao i između clustered i non-clustered organizacije, ključno je za dizajniranje optimizovanih baza i upita koji se izvršavaju brzo i pouzdano.



Ovaj dijagram prikazuje hijerarhijsku strukturu B-Tree indeksa u InnoDB mehanizmu — sa jasno označenim korenskim čvorom, unutrašnjim čvorovima i listovima, kao i strelicama koje vizualizuju tok pretrage.

* 1. **MyISAM: B-Tree indeksiranje i odvojena struktura za podatke i indeks**

Pored InnoDB-a, jedan od tradicionalnih skladišnih mehanizama u MySQL-u jeste **MyISAM**. Iako je danas manje korišćen u produkcionim sistemima zbog nedostatka podrške za transakcije i referencijalni integritet, MyISAM i dalje ima značajnu ulogu u razumevanju osnovnih principa indeksiranja i organizacije podataka u MySQL-u.

* 1. **B-Tree indeksiranje u MyISAM-u**

MyISAM koristi **B-Tree strukture** za implementaciju svih tipova indeksa — uključujući primarne, jedinstvene i obične indekse. B-Tree omogućava efikasnu pretragu, umetanje i brisanje podataka, uz očuvanje sortiranja. Svaki čvor u stablu sadrži više ključeva i pokazivača ka podčvorovima, čime se omogućava logaritamska složenost pretrage.

Za razliku od InnoDB-a, gde primarni indeks ujedno određuje fizički raspored podataka (clustered indeks), u MyISAM-u svi indeksi su **non-clustered**. To znači da se podaci ne nalaze unutar indeksne strukture, već se indeks koristi kao pokazivač ka fizičkoj lokaciji podataka u posebnom fajlu.

* 1. **Odvojena struktura za podatke i indeks**

Jedna od ključnih karakteristika MyISAM-a jeste **fizička separacija podataka i indeksa**. Svaka MyISAM tabela se fizički sastoji od dva (ili tri) fajla:

* **.MYD fajl (MyISAM Data)** – sadrži stvarne podatke iz tabele.
* **.MYI fajl (MyISAM Index)** – sadrži sve indeksne strukture.
* (Opcionalno) **.frm fajl** – sadrži definiciju tabele (u starijim verzijama MySQL-a).

Ova odvojenost omogućava da se indeksne strukture brzo učitavaju i obrađuju, bez potrebe za direktnim pristupom podacima. Kada se izvrši upit, MySQL koristi indeks iz .MYI fajla da pronađe fizičku lokaciju podatka u .MYD fajlu, čime se ubrzava pristup.

* 1. **Prednosti i ograničenja**

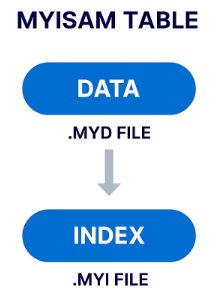
**Prednosti:**

* Brzo učitavanje indeksnih struktura.
* Efikasno izvođenje SELECT upita nad statičnim podacima.
* Manja potrošnja memorije kod jednostavnih tabela.

**Ograničenja:**

* Nema podrške za transakcije (ACID).
* Nema zaključavanja na nivou reda — koristi se zaključavanje cele tabele.
* Nema podrške za strane ključeve i referencijalni integritet.
* Veća osetljivost na oštećenje fajlova pri naglom prekidu rada.

U zaključku, MyISAM predstavlja jednostavan, ali ograničen skladišni mehanizam koji koristi B-Tree indeksiranje i fizičku separaciju podataka i indeksa. Iako je danas zamenjen naprednijim engine-ima poput InnoDB-a, razumevanje njegove interne organizacije pruža korisnu osnovu za razumevanje principa indeksiranja u MySQL-u.

****

Ovaj dijagram vizualno prikazuje kako MyISAM skladišni mehanizam u MySQL-u razdvaja podatke (.MYD fajl) od indeksa (.MYI fajl), uz strelicu koja pokazuje kako indeks vodi do fizičke lokacije podataka.

* 1. **Kako MySQL održava statistiku indeksa**

Statistika indeksa u MySQL-u predstavlja skup metapodataka koje baza koristi za procenu efikasnosti izvršavanja SQL upita. Ove statistike omogućavaju optimizatoru upita da donese informisane odluke o tome koji indeks koristiti, kojim redosledom izvršavati JOIN operacije i kako filtrirati podatke na najefikasniji način. Bez tačnih i ažurnih statistika, čak i dobro dizajnirani indeksi mogu biti ignorisani, što dovodi do neoptimalnog izvršavanja upita.

**Šta obuhvata statistika indeksa?**

Statistika indeksa obično uključuje sledeće informacije:

* **Broj različitih vrednosti (cardinality)** u indeksiranoj koloni
* **Ukupan broj redova** u tabeli
* **Distribucija vrednosti** u indeksu (ravnomerna vs. koncentrisana)
* **Veličina indeksa** u memoriji i na disku
* **Dubina B-Tree strukture** (broj nivoa od korena do listova)

Ovi podaci se čuvaju interno i koriste se prilikom generisanja plana izvršavanja upita (query execution plan).

**Kako se statistika prikuplja i osvežava?**

MySQL automatski prikuplja statistiku indeksa prilikom:

* **Kreiranja indeksa**
* **Velikih promena u tabeli** (npr. masovni INSERT, DELETE, UPDATE)
* **Ručno pokrenute analize** pomoću komande ANALYZE TABLE

**Primer:**



Ova komanda forsira MySQL da osveži statistiku za sve indekse u tabeli „korisnici“, čime se optimizatoru omogućava da koristi najnovije podatke prilikom planiranja upita.

**Gde se koristi statistika indeksa?**

Statistika indeksa se koristi u tzv. **query optimizeru**, koji odlučuje:

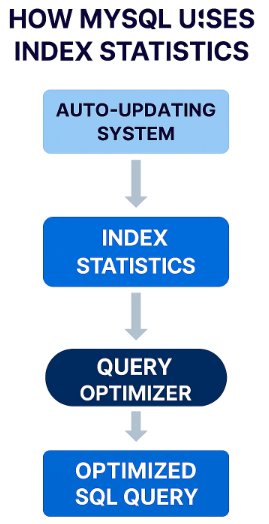
* Da li koristiti indeks ili izvršiti full table scan
* Koji indeks je najefikasniji za dati upit
* Redosled izvršavanja više JOIN operacija
* Da li koristiti pokriveni indeks (covering index) za čitanje podataka bez pristupa tabeli

Na osnovu statistike, MySQL može odlučiti da ignoriše indeks ako proceni da je broj pogodaka mali ili da je indeks loše raspoređen.

**Potencijalni problemi sa statistikom**

* **Zastarela statistika**: Ako se tabela često menja, a statistika se ne osvežava, optimizator može doneti pogrešne odluke.
* **Netipična distribucija podataka**: Ako indeks sadrži mnogo ponavljajućih vrednosti, statistika može biti neprecizna.
* **Automatsko osvežavanje nije uvek pouzdano**: U nekim verzijama MySQL-a, automatsko ažuriranje statistike može biti odloženo ili neprecizno, pa se preporučuje ručno pokretanje ANALYZE TABLE.

Statistika indeksa je nevidljiva ali ključna komponenta performansi baze podataka. Njeno pravilno održavanje omogućava da se indeksi koriste optimalno, da se upiti izvršavaju efikasno i da se izbegnu nepotrebna skeniranja tabela. U kontekstu velikih i dinamičnih baza, redovno osvežavanje statistike predstavlja važan korak u administraciji i optimizaciji sistema.



Ovaj dijagram vizualno prikazuje kako MySQL koristi statistiku indeksa za optimizaciju upita: od automatskog sistema koji prikuplja podatke, preko indeksne statistike, zatim kroz query optimizer, sve do generisanja optimizovanog SQL plana.

* 1. **Fragmentacija i ažuriranje indeksa**

Indeksi u bazama podataka nisu statične strukture — oni se dinamički menjaju kako se podaci u tabelama dodaju, brišu ili ažuriraju. Tokom vremena, ove promene mogu dovesti do **fragmentacije indeksa**, što negativno utiče na performanse upita, efikasnost pretrage i potrošnju resursa. Razumevanje procesa fragmentacije i načina ažuriranja indeksnih struktura ključno je za održavanje optimalnog rada baze podataka.

**Šta je fragmentacija indeksa?**

Fragmentacija indeksa označava stanje u kojem se fizička organizacija indeksne strukture udaljava od optimalne — tj. kada se podaci u B-Tree stablu rasporede neefikasno zbog čestih umetanja, brisanja i ažuriranja. To može dovesti do:

* **Povećanja dubine stabla** (više nivoa između korena i listova)
* **Praznih ili polupraznih čvorova**
* **Neefikasnog korišćenja memorije**
* **Usporavanja pretrage i navigacije kroz indeks**

U InnoDB mehanizmu, fragmentacija se najčešće javlja u sekundarnim indeksima, jer se podaci ne skladište direktno u indeksu, već se koristi pokazivač ka primarnom ključu.

**Kada i zašto dolazi do fragmentacije?**

Fragmentacija se javlja u sledećim situacijama:

* **Masovno brisanje podataka**: oslobađaju se čvorovi u indeksu, ali se ne reorganizuju automatski.
* **Umetanje podataka sa vrednostima koje narušavaju postojeći redosled**: indeks mora da se proširi i balansira.
* **Česta ažuriranja vrednosti u indeksiranim kolonama**: može doći do premeštanja podataka unutar stabla.

Ove promene dovode do toga da indeks zauzima više prostora nego što je potrebno i da se pretraga vrši sporije.

**Ažuriranje i održavanje indeksa**

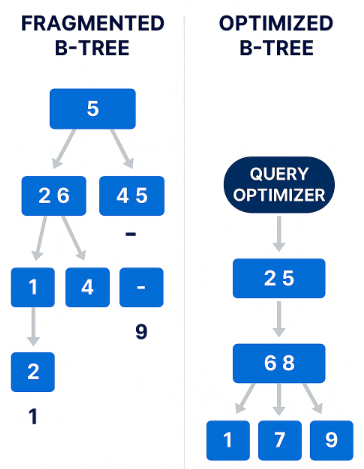
MySQL automatski ažurira indeksne strukture prilikom svake promene podataka u indeksiranim kolonama. Međutim, automatsko ažuriranje ne rešava problem fragmentacije. Da bi se indeks reorganizovao i optimizovao, potrebno je izvršiti eksplicitne operacije:

* **OPTIMIZE TABLE** Ova komanda rekreira tabelu i sve njene indekse, čime se uklanja fragmentacija i smanjuje veličina fajlova.

****

* **ALTER TABLE ... ENGINE=InnoDB** Alternativno, promena engine-a (čak i na isti) može izazvati rekonstrukciju indeksa.
* **DROP INDEX / ADD INDEX** Ručno uklanjanje i ponovno kreiranje indeksa može biti korisno kod ekstremne fragmentacije.
* **ANALYZE TABLE** Osvežava statistiku indeksa, ali ne rešava fizičku fragmentaciju.

U zaključku, fragmentacija indeksa je prirodan rezultat dinamičnog rada sa podacima, ali ako se ne kontroliše, može dovesti do ozbiljnih problema sa performansama. Redovno održavanje i optimizacija indeksnih struktura predstavlja važan deo administracije baze podataka, naročito u sistemima sa velikim brojem transakcija i promena.



Prikazuje vizuelnu razliku između **fragmentiranog** i **optimizovanog B-Tree indeksa** u MySQL-u. Na levoj strani vidiš nebalansirane grane, prazne čvorove i neefikasnu raspodelu podataka, dok desna strana prikazuje uredno balansiranu strukturu sa ravnomernim grananjem i punim listovima — idealno stanje nakon optimizacije.

1. **KORIŠĆENJE INDEKSA U UPITIMA**

Indeksi u MySQL-u imaju ključnu ulogu u optimizaciji izvršavanja SQL upita. Njihova osnovna funkcija je da omoguće brži pristup podacima, smanjujući potrebu za sekvencijalnim pretraživanjem cele tabele. Kada se upit odnosi na kolonu koja je indeksirana, MySQL koristi indeksnu strukturu da pronađe relevantne redove efikasno, što značajno poboljšava performanse baze podataka, naročito kod velikih tabela.

* 1. **Načini na koje MySQL koristi indekse**

Indeksi u MySQL-u nisu samo statične strukture za ubrzavanje pretrage — oni se aktivno koriste u različitim vrstama SQL upita kako bi se poboljšale performanse, smanjilo vreme izvršavanja i optimizovalo korišćenje resursa. U nastavku su prikazani najvažniji načini na koje MySQL koristi indekse tokom obrade upita.

* 1. **WHERE uslovi**

Jedan od najčešćih scenarija u kojem se koristi indeks jeste filtriranje podataka pomoću **WHERE** klauzule. Kada se uslov odnosi na kolonu koja je indeksirana, MySQL koristi indeksnu strukturu da pronađe redove koji zadovoljavaju kriterijum, bez potrebe za pregledom cele tabele.

**Primer:**



Ako kolona grad ima indeks, baza koristi B-Tree pretragu da brzo pronađe sve korisnike iz Leskovca.

Indeks se koristi i kod složenijih uslova, kao što su:

* WHERE godina BETWEEN 2010 AND 2020
* WHERE status IN ('aktivan', 'neaktivan')

Važno je da uslov bude direktno nad indeksiranom kolonom i da ne koristi funkcije koje bi onemogućile korišćenje indeksa (npr. WHERE UPPER(ime) = 'BOJAN').

* 1. **JOIN operacije**

Kod **JOIN** operacija, indeksi omogućavaju brzo pronalaženje povezanih redova između dve ili više tabela. Kada se tabele povezuju preko kolona koje su indeksirane, MySQL koristi indeks da pronađe odgovarajuće redove u drugoj tabeli, čime se izbegava kombinovanje svakog reda sa svakim.

**Primer:**



Ako korisnik\_id u porudzbine i id u korisnici imaju indekse, JOIN se izvršava efikasno, uz minimalno opterećenje.

Indeksi su posebno korisni kod višestrukih JOIN-ova, gde se redosled izvršavanja operacija određuje na osnovu statistike indeksa.

* 1. **ORDER BY i GROUP BY**

Indeksi se koriste i za **sortiranje** (ORDER BY) i **grupisanje** (GROUP BY) podataka. Kada se ove operacije vrše nad indeksiranom kolonom, MySQL može da koristi već sortirane vrednosti iz indeksa, čime se izbegava dodatno sortiranje u memoriji ili na disku.

**Primer ORDER BY:**

****

**Primer GROUP BY:**



Ako kolona grad ima indeks, grupisanje se vrši brže jer se vrednosti već nalaze u organizovanom obliku.

**4.5 LIKE sa prefiksom**

Indeksi se mogu koristiti i kod **LIKE** operatora, ali samo kada se pretraga vrši sa **prefiksom** — tj. kada se traženi tekst nalazi na početku vrednosti.

**Primer (koristi indeks):**

****

U ovom slučaju, MySQL može da koristi B-Tree indeks nad kolonom ime, jer se pretraga odnosi na početak stringa.

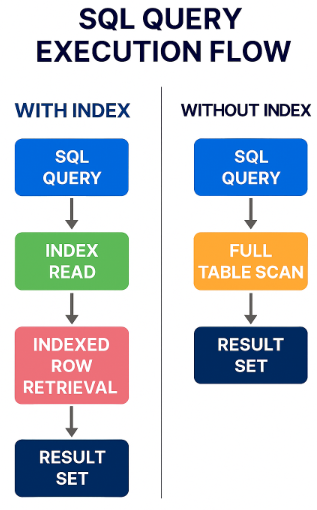
**Primer (ne koristi indeks):**

****

Kada se koristi % na početku, indeks se ne može koristiti jer baza mora da pregleda sve vrednosti u koloni.

Za naprednu pretragu teksta, preporučuje se korišćenje **FULLTEXT indeksa**, koji omogućava pretragu po ključnim rečima i frazama.

MySQL koristi indekse na više načina kako bi optimizovao izvršavanje upita. Pravilno dizajnirani indeksi omogućavaju efikasno filtriranje, povezivanje, sortiranje i pretragu podataka, dok loše postavljeni indeksi mogu biti zanemareni od strane optimizatora. Razumevanje ovih mehanizama ključno je za dizajniranje performantnih baza i aplikacija.



Vizualno prikazuje razliku između efikasnog pristupa podacima korišćenjem indeksa (levo) i sporijeg pristupa putem full table scan-a (desno).

1. **OPTIMIZACIJA UPITA POMOĆU INDEKSA**

U radu sa relacionim bazama podataka, brzina i efikasnost izvršavanja upita predstavljaju ključne faktore za performanse sistema. Kako se obim podataka u tabelama povećava, neophodno je primeniti mehanizme koji omogućavaju da se upiti izvršavaju brzo i uz minimalno opterećenje sistema. Jedan od najvažnijih mehanizama za optimizaciju upita u MySQL-u jeste korišćenje indeksa.

Indeksi omogućavaju da se podaci pronađu, filtriraju, sortiraju i povežu znatno brže nego pri sekvencijalnom pregledu cele tabele. Umesto da baza pregleda svaki red pojedinačno, koristi se strukturirana pretraga kroz indeksnu strukturu (najčešće B-Tree), čime se vreme izvršavanja smanjuje sa linearne na logaritamsku složenost.

**5.1 Korišćenje EXPLAIN za analizu upita**

U procesu optimizacije SQL upita, važno je ne samo definisati indekse, već i proveriti da li ih MySQL zaista koristi prilikom izvršavanja. Za tu svrhu koristi se komanda **EXPLAIN**, koja prikazuje plan izvršavanja upita — tj. kako MySQL pristupa podacima, koje indekse koristi, koliko redova procenjuje da će pregledati i kojim redosledom obrađuje tabele.

EXPLAIN omogućava da se identifikuju neefikasni upiti, nepotrebni full table scan-ovi i da se potvrdi da indeksiranje daje očekivane performanse.

**Sintaksa i primer:**



Ova komanda prikazuje tabelarni rezultat sa više kolona koje opisuju kako će se upit izvršiti. Najvažnije kolone u izlazu su:

* **id** – identifikator upita (korisno kod složenih JOIN-ova)
* **select\_type** – tip upita (npr. SIMPLE, PRIMARY, SUBQUERY)
* **table** – ime tabele koja se obrađuje
* **type** – tip pristupa podacima (npr. ref, range, ALL)
* **key** – ime indeksa koji se koristi
* **key\_len** – dužina ključa u bajtovima
* **rows** – procenjeni broj redova koje će MySQL pregledati
* **Extra** – dodatne informacije (npr. Using index, Using where, Using filesort)

**Tumačenje rezultata:**

* Ako je vrednost u koloni type npr. ref ili range, to znači da se koristi indeks i da je pristup efikasan.
* Ako je vrednost ALL, to označava full table scan — što je znak da indeks nije iskorišćen.
* Kolona key pokazuje tačno koji indeks je korišćen, dok key\_len daje tehnički detalj o dužini ključa.
* Broj u koloni rows pomaže da se proceni koliko podataka će biti obrađeno — manji broj znači brži upit.

**Primer analize**

**Za upit:**

****

Ako je kreiran složeni indeks nad kolonama knjiga\_id i datum\_izdavanja, očekuje se da EXPLAIN prikaže:

* key: idx\_knjiga\_datum
* type: ref ili range
* rows: mali broj (npr. 1–2)
* Extra: Using where

To potvrđuje da se indeks koristi i da je upit optimizovan.

Komanda EXPLAIN je neophodan alat za svakog administratora baze i programera koji želi da osigura da se indeksi koriste pravilno. Kroz analizu plana izvršavanja, moguće je identifikovati uska grla, prilagoditi strukturu upita i poboljšati performanse sistema. Redovno korišćenje EXPLAIN-a predstavlja najbolju praksu u radu sa MySQL bazama podataka.

**5.2 Composite indeksi i redosled kolona**

Složeni (composite) indeksi predstavljaju indekse koji obuhvataju više kolona istovremeno. Njihova primena je posebno korisna kod upita koji filtriraju podatke na osnovu kombinacije više kriterijuma. Međutim, da bi se composite indeks efikasno koristio, neophodno je da se redosled kolona u indeksu poklapa sa redosledom u kojem se te kolone pojavljuju u upitu.



**Ovaj indeks će se koristiti u upitu:**



Ali neće biti iskorišćen u potpunosti ako se filtrira samo po datum\_izdavanja, jer knjiga\_id dolazi prvi u indeksu. Zbog toga je redosled kolona u composite indeksu od suštinskog značaja — indeks se koristi od leve ka desnoj koloni, i svaka sledeća kolona se koristi samo ako su prethodne uključene u uslov upita.

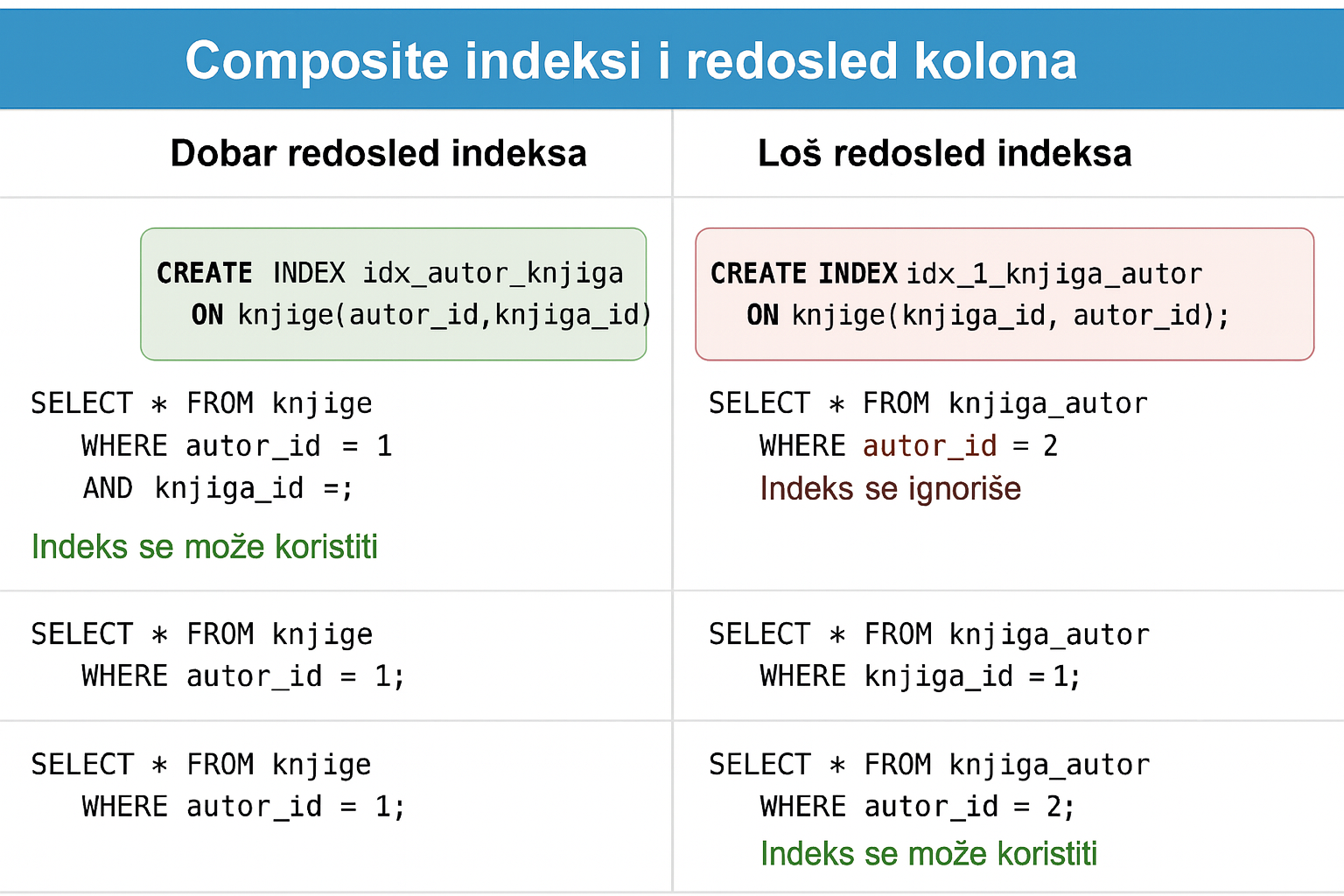
**5.3 Preporuke za smanjenje nepotrebnog indeksiranja**

Iako indeksi poboljšavaju performanse, prekomerno indeksiranje može imati negativne posledice: povećanu potrošnju memorije, sporije operacije umetanja i ažuriranja, kao i komplikovanije održavanje baze. Zbog toga je važno primenjivati indekse selektivno i strateški.

**Preporuke:**

* **Analiziraj najčešće upite**: Kreiraj indekse samo za kolone koje se redovno koriste u WHERE, JOIN, ORDER BY ili GROUP BY klauzulama.
* **Izbegavaj indeksiranje kolona sa malim brojem različitih vrednosti** (npr. status sa vrednostima 'aktivan' i 'neaktivan') — indeks u tim slučajevima često ne donosi značajno ubrzanje.
* **Ne indeksiraj sve strane ključeve automatski**: Ako se kolona koristi samo za referencijalni integritet, a ne za pretragu, indeks možda nije potreban.
* **Koristi composite indekse umesto više pojedinačnih**: Ako se upiti filtriraju po više kolona istovremeno, composite indeks je efikasniji od više zasebnih indeksa.
* **Redovno koristi EXPLAIN**: Analiziraj plan izvršavanja upita da bi proverio da li se indeks koristi. Ako se ne koristi, razmisli o njegovom uklanjanju.
* **Prati veličinu indeksnih fajlova**: U MyISAM mehanizmu, indeks se čuva odvojeno (.MYI fajl), pa se može pratiti rast fajla kao indikator prekomernog indeksiranja.

Composite indeksi su moćan alat za optimizaciju složenih upita, ali zahtevaju pažljivo planiranje redosleda kolona. Istovremeno, racionalno indeksiranje doprinosi stabilnosti i efikasnosti baze, sprečavajući nepotrebno opterećenje sistema. Pravilna strategija indeksiranja podrazumeva balans između performansi i održivosti.



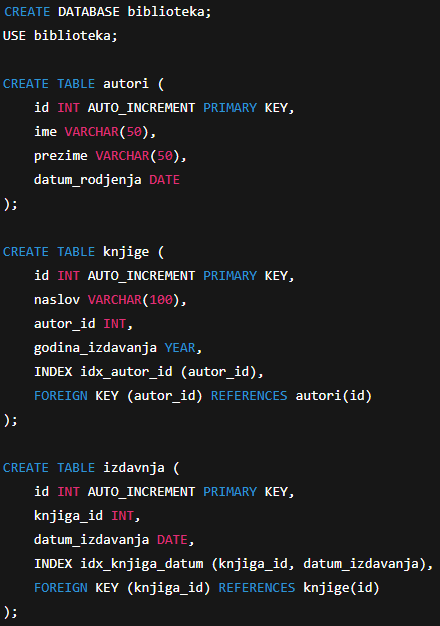
Ova tabela prikazuje uporedno **dobar** i **loš** redosled kolona u composite indeksima, uz konkretne SQL primere i napomene da li se indeks koristi ili ignoriše.

1. **PRAKTIČAN DEO-DEMONSTRACIJA U MYSQL-U**

U ovom delu rada prikazana je praktična primena indeksiranja u MySQL-u kroz kreiranje baze podataka, definisanje tabela sa indeksima, unos podataka i analiza izvršavanja SQL upita. Cilj je da se demonstrira kako se indeksi implementiraju i koriste u skladu sa internom strukturom MySQL-a, posebno B-Tree organizacijom, koja omogućava efikasnu pretragu i filtriranje podataka.

* 1. **Kreiranje baze i tabela**

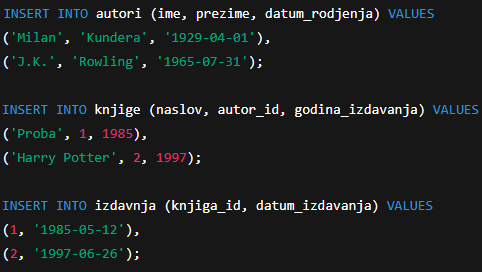
Kreirana je baza podataka **biblioteka**, sa tri povezane tabele: autori, knjige i izdavnja. Indeksi su dodati na kolone koje se koriste za pretragu i povezivanje, u skladu sa principima interne strukture B-Tree indeksa.

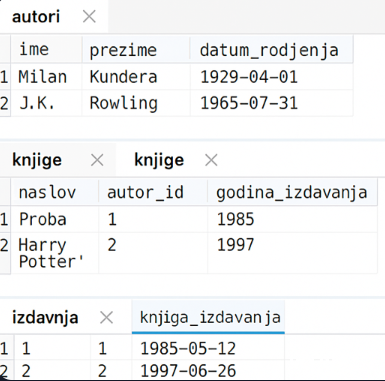


U tabeli knjige koristi se jednostavan B-Tree indeks nad kolonom autor\_id, dok tabela izdavnja koristi **složeni (composite) indeks** nad kolonama knjiga\_id i datum\_izdavanja, što omogućava efikasnu pretragu po kombinaciji vrednosti.

* 1. **Popunjavanje tabela podacima**

**Dodavanje test podataka omogućava demonstraciju rada indeksa u realnim upitima:**





Prikaz svih tabela iz baze u MySQL Workbench-a.

* 1. **Korišćenje indeksa u upitima**

**Sledeći upiti demonstriraju kako se indeksi koriste u skladu sa internom strukturom MySQL-a:**

* **Pretraga po autoru** (koristi indeks idx\_autor\_id):

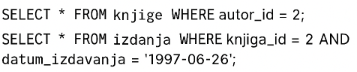
****

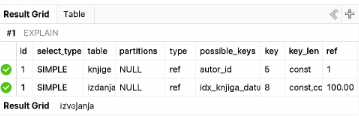
* **Sortiranje po godini izdavanja**:

****

* **Korišćenje composite indeksa** idx\_knjiga\_datum:

****





U svim slučajevima, MySQL koristi B-Tree strukturu da brzo pronađe redove bez potrebe za sekvencijalnim pregledom cele tabele.

* 1. **Analiza upita pomoću EXPLAIN**

Komanda EXPLAIN omogućava da se proveri da li MySQL koristi indeks i kako se upit izvršava u pozadini.

**Primer:**

****

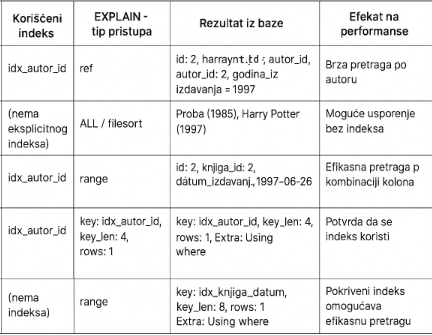
**Tumačenje ključnih kolona u izlazu:**

* key: ime indeksa koji se koristi (npr. idx\_autor\_id)
* key\_len: dužina ključa u bajtovima (npr. 4 za INT)
* rows: procenjeni broj redova koje MySQL mora da pregleda
* type: tip pristupa (ref, range, ALL)
* Extra: dodatne informacije (Using index, Using where)

Ako se u koloni key pojavi naziv indeksa, to znači da je indeks uspešno iskorišćen. Tip ref označava efikasnu pretragu kroz B-Tree strukturu, dok ALL ukazuje na full table scan — što je znak da indeks nije iskorišćen.

Prikazani primer pokazuju kako se indeksi u MySQL-u definišu i koriste u skladu sa njihovom internom strukturom. Kroz jednostavnu bazu podataka i konkretne upite, jasno se vidi da indeksiranje omogućava bržu pretragu, efikasnije izvršavanje upita i optimizaciju performansi. Korišćenjem komande EXPLAIN, moguće je potvrditi da se indeks koristi i da je upit pravilno optimizovan.

* 1. **Rezultati testiranja na MySQL client-u**



Ova tabela tj. grafički prikaz koju smo pripremili predstavlja **kompaktni pregled praktičnog rada** u MySQL-u i jasno pokazuje kako se indeksi koriste u realnim upitima, kako ih MySQL interpretira kroz EXPLAIN analizu, i kakav je efekat na performanse baze.

1. **ZAKLJUČAK**

Indeksi predstavljaju jedan od najvažnijih mehanizama za optimizaciju rada sa relacionim bazama podataka. Njihova osnovna funkcija je da ubrzaju pristup podacima, smanje opterećenje sistema i omoguće efikasno izvršavanje upita. Kroz teorijski pregled i praktičnu demonstraciju u MySQL okruženju, jasno je prikazano kako se indeksi kreiraju, koriste i analiziraju — od jednostavnih do složenih (composite) struktura.

Korišćenjem indeksa, MySQL može da izbegne sekvencijalno pretraživanje tabela, koristi B-Tree strukture za logaritamsku pretragu, efikasno izvršava JOIN operacije, sortiranje (ORDER BY), grupisanje (GROUP BY) i filtriranje (WHERE). Analiza upita pomoću komande EXPLAIN omogućava uvid u plan izvršavanja i potvrđuje da li se indeks koristi u praksi.

* 1. **Preporuke za efikasno indeksiranje u MySQL-u**
* Indeksirati kolone koje se često koriste u filtriranju, povezivanju i sortiranju.
* Koristiti composite indekse kada se upiti oslanjaju na više kolona istovremeno, uz pažljiv izbor redosleda.
* Izbegavati prekomerno indeksiranje koje može usporiti operacije umetanja i ažuriranja.
* Redovno analizirati upite pomoću EXPLAIN da bi se identifikovali neefikasni obrasci.
* Pratiti veličinu indeksnih fajlova i uticaj na performanse sistema.
  1. **Smernice za dalji razvoj**
* **Optimizacija**: Uvođenje dinamičkog praćenja performansi upita i automatske preporuke za indeksiranje.
* **Održavanje**: Periodično korišćenje komandi kao što su OPTIMIZE TABLE i ANALYZE TABLE za uklanjanje fragmentacije i osvežavanje statistike.
* **Monitoring**: Implementacija alata za praćenje korišćenja indeksa, identifikaciju neiskorišćenih indeksa i analizu trendova u upitima.

U budućem radu, posebna pažnja može biti posvećena naprednim tipovima indeksa (npr. FULLTEXT, SPATIAL), kao i integraciji indeksiranja sa sistemima za automatsko skaliranje i distribuciju podataka. Time se otvara prostor za dublje razumevanje performansi baza u kompleksnim aplikacionim okruženjima.

1. **REFERENCE**
2. **Zvanična dokumentacija MySQL** MySQL Reference Manual. Dostupno na: <https://dev.mysql.com/doc/>
3. **Schwartz, B., Zaitsev, P., Tkachenko, V.** High Performance MySQL: Optimization, Backups, and Replication. O'Reilly Media, 3rd Edition, 2012.
4. **Online izvori** Članci, blogovi i tutorijali o indeksiranju, optimizaciji upita i analizi performansi u MySQL-u. (npr. Percona blog, DigitalOcean tutorials, MySQL Performance Blog)
5. **Maesaroh, S., Gunawan, H., Lestari, A., Tsaurie, M.S.A., & Fauji, M.** Query Optimization in MySQL Database Using Index. International Journal of Cyber and IT Service Management (IJCITSM), Vol. 2, No. 2, 2022, str. 104–110.

**HVALA NA PAŽNJI 😊**