UNIVEZITET U NIŠU

ELEKTRONSKI FAKULTET

Seminarski rad

Interna struktura i organizacija indeksa u MySql bazi podataka

Predmet: Sistemi za upravljanje bazama podataka

Mentor: Aleksandar Stanimirović

Student: Marija Stojanović 1034

Niš, april 2020. god

Sadržaj

1 Uvod

Indeks u bazi podataka predstavlja strukturu podataka koja poboljšava brzinu izvršavanja operacija za pretragu i dobijanje informacija iz baze podataka.[12]Indeksi se koriste kako bi se brže locirao podatak bez potrebe da se pretraži svaki red u tabeli baze podataka svaki put kad se pristupi bazi. Ipak postoji dodatna cena koja se plaća korišćenjem indeksa koja se odnosi na dodatne upise i dodatan prostor za skladištenje kako bi se upravljalo samom indeksnom strukturom podataka.[12]

Indeksi mogu biti kreirani koristeći jednu ili više kolona iz tabele baze podataka pružajući osnovu sa i za brza slučajna pretraživanja kao i efikasan pristup uređenim rekordima baze.

Indeks predstavlja kopiju selektovanih kolona podataka iz tabele koji se naziva ključ baze podataka ili jednostavno ključ, a koji može biti pretraživan vrlo efikasno i uključuje i adresu bloka diska niskog nivoa ili direkrnu vezu do celog niza podataka koji su kopirani iz neke baze podataka.

Slikovito indekse možemo predstaviti kao početak knjige na kome se nalazi sadržaj gde je svaki naslov određen brojem stranice, a i kao kraj njige koji sadrži rečnik pojmova koji se nalaze u knjizi kao i stranice na kojima se ti pojmovi nalaze.

U poglavlju 2 biće objašnjena osnovna ideja i funkcionalnosti indeksa. Biće predstavljeni clustered, non-clustered indeksi, kao i parcijalni indeksi i indeksi nad jednom i više kolona.

U poglavlju 3 biće objašnjena struktura podataka kojom su predstavljeni indeksi. Biće konkretnije objašnjeni indeksi predstavljeni B-stablom, indeksi predstavljeni R-stablom, kao i indeksi predstavljeni Hash tabelom.

U poglavlju 4 biće govora o pet osnovnih tipova indeksa u MySQL-u.

U poglavlju 5 biće govora o tome koje vrste tabela za skladištenje podataka u MySQL-u podržavaju koje tipove indeksna i koje indeksne strukture.

U poglavlju 6 biće predstavljeni načini za kreiranje, brisanje i preimenovanje indeksa u MySQL-u.

2 Osnovni koncepti indeksiranja

Da bismo mogli da razumemo kako MySQL koristi indekse, moramo prvo razumeti osnovne funkcije i karakteristike indeksa. Jednom kada se razumeju osnovne karakteristike indeksa, može se početi sa pravljenjem intelignetnih poteza koji se tiču korišćenja indeksa.[13]

2.1 Ideja indeksiranja

Kako bismo razumeli šta indeksi omogućavaju MySQL-u, najbolje je da pogledamo kako MySQL radi pri odgovaranju na upite. Zamislimo da je phone_book table a koja sadrži ceo spisak brojeva u Klaiforniji sa oko 35 miliona ulaza. Takođe pretpostavićemo da rekordi u tabeli nisu sortirani. Uzmimo u obzir sledeći upit:[13]

```
SELECT * FROM phone_book WHERE last_name = 'Zawodny'
```

Slika 1: SQL upit

Bez bilo kakvog sortiranog indeksa MySQL mora pročitati sve rekorde iz tabele phone_book i za svaki uporedi polje last_name sa stringom “Zawody” kako bi odredio one koji se poklapaju. Jasno je da ovakav pristup nije efikasan. Kako raste broj rekorda tako se povećava vreme potrebno da se traženi rekord pronađe. U računarstvu se to zove O(n) problem.[13]

Ako pogledamo kako funkcioniše stvarna knjiga svih telefonskih brojeva u nekoj državi, setićemo se da svi zapravo znamo kako bismo našli na brz način nekog ko se zove Zawodny. Imali bismo na kraju ove knjige indeks sa svim imenima, gde su ona sortirana po alfabetu, tako da bismo našli prvo a onda i drugo slovo imena i odatle počeli pretragu. [13]

Mnoge knjige imaju indeks pojmova koji se nalazi na kraju knjige, a u kome imamo pojmove koji su uređeni po alfabetu i pored svakog pojma se nalazi stranica na kojoj se taj pojam nalazi. Slični ovima su i indeksi kod baza podataka. Kod baza podataka možemo izabrati da kreiramo indekse nad nekim kolonama u bazi podataka. Ako uzmemo u obzir prethodni primer, možemo kreirati indeks nad kolonom last_name i tako da ubrzamo pretragu telefonskih brojeva:[13]

```
ALTER TABLE phone_book ADD INDEX (last_name)
```

Slika 2: Naredba za kreiranje indeksa nad kolonom last_name

Navedenom naredbom MySQL će kreirati uređenu listu svih polja last_name u phone_book tabeli. Uz svako ime biće pridodate pozicije rekorda koji se poklapaju s njim baš ako što izgleda i indeks na kraju bilo koje knjige koji sadrži brojeve strana na kojima se pojmovi nalaze.[13]

Iz ugla servera baze podataka, idneksi postoje kako bi baza podataka mogla brzo da eliminiše redove iz rezultata tokom izvršenja upita. Bez indeksa ovo ne bi bilo moguće tako da bi MySQL morao da pređe svaki red u tabeli. Ne samo da bi došlo do gubitka previše vremena već bi se moralo izvršiti previše I/O operacija na disku pa bi se tako mogla dodatno preoteretiti keš memorija.[13]

Svakako, indekse nije preporučljivo formirati nad svakom kolonom u tabeli. Indeksi predstavljaju kompromis između vremena i prostora. Naime ako imamo formirane indekse onda nam je za svaku INSERT, UPDATE i DELETE naredbu nad bazom podataka potreban dodatni CPU i prostor na disku koji zahteva pamćenje i rad sa indeksima.[13]

U većem delu MySQL dokumentacije termin *indeks* zamenjuje se terminom *ključ*. Reći da je last_name ključ tabele phone_book je isto kao i reći da je last_name indeksirana kolona phone_book tabele.[13]

2.2 Parcijalni indeksi

Kao šro je već rečeno indeksi zahtevaju dodatni prostor jer je potrebna memorija u kojoj će se oni pamtiti. Na sreću MySQL nam omogućava da imamo kontrolu nad tim koliko prostora će se koristiti za smeštanje indeksa. Ako pogledamo prethondno pomenutu tabelu phone_book tabelu sa 2 biliona redova, videćemo da da indeksiranje kolone last_name može zahtevati dosta memorijskog prostora. Umesto indeksiranja cele kolone last_name MySQL nam dopušta da indeksiramo samo prva 4 bajta:[13]

```
ALTER TABLE phone_book ADD INDEX (last_name(4))
```

Slika 3: Naredba za kreiranje parcijalnog indeksa

2.3 Indeksi nad jednom ili više kolona

Indeksi se mogu praviti nad jednom ili više kolona. Činjenica je da u MySQL-u verzije 8 indeks može sadržati do 16 kolona, ali uglavnom nikada nije potrebno toliko mnogo kolona u indeksu. Da li će se koristiti jedna ili više kolona za kreiranje indeksa zavisi od tipa podataka koji je potrebno pretraživati.[7]

Primera radi, možemo imati tabelu sa kupcima koja sadrži kolone first_name i last_name. Ako se ikada pretražuju vrednosti kolone last_name u tom slučaju indeks po koloni last_name bi bio odgovarajući. Ako pak bude bilo potrebno da se pretraže i first_name i last_name u jednom upitu, onda je najbolje napraviti indeks nad više kolona tj. nad kolonama first_name i last_name.[7]

Generalno indeksi koji se sastoje od više kolona koriste se ukoliko se želi povećati brzina izvršavanja upita koji testiraju sve kolone u indeksu ili samo prvu kolona, ili prvu i drugu kolonu ili prvu, drugu i treću kolonu i tako dalje. Ako upit koji se izvršava testira jednu kolonu, onda je za to najbolje koristiti indeks nad jednom kolonom.[7]

2.4 Clustered indeksi

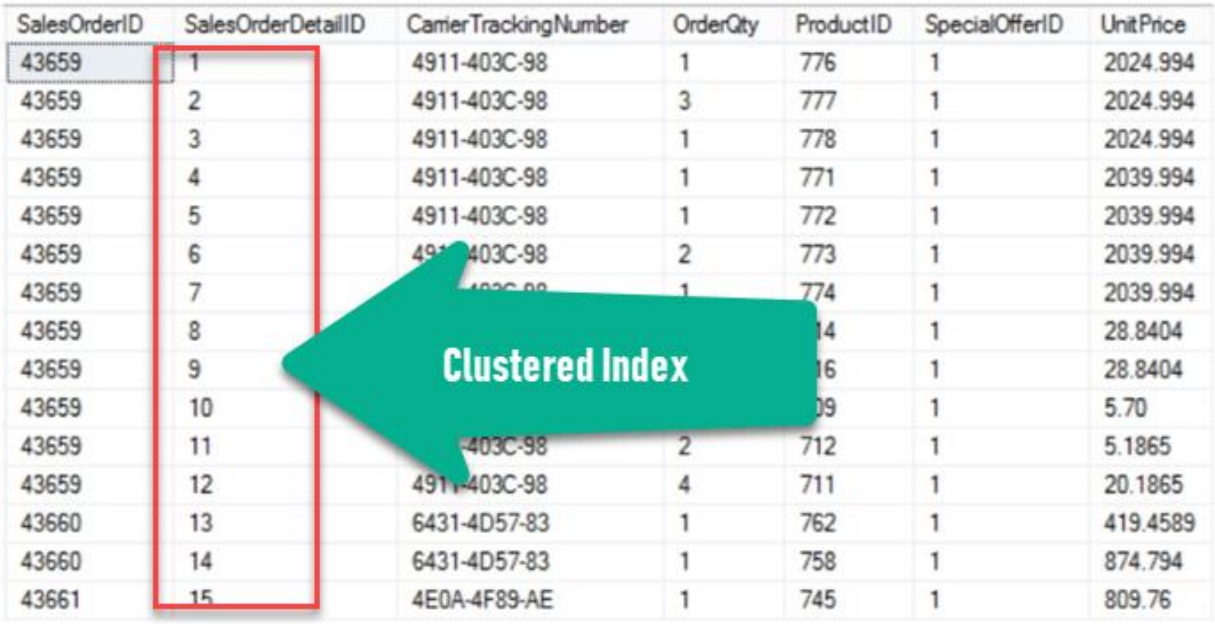
Svaka tabela poseduje specijalni indeks poznat kao clustered indeks gde su podaci o redovima date tabele smešteni. Clustered indeks zapravo je predstavljen u vidu zasebne tabele. [8] Kod ovog tipa indeksa, redovi podataka se čuvaju na disku po istom rasporedu kao što su postavljeni u tabeli. Zbog toga, možemo imati samo jedan clustered indeks po tabeli.[7] Sinonim za clustered indeks je primarni ključ.[9]

Clustered indeksi su generanlno brži za čitanje sve dok možemo dobiti sve potrebne informacije iz indeksa i ne moramo uzimati podatke iz originalne tabele. Za upis, clustered indeks može zahtevati više vremena, pogotovo ako novi podaci zahtevaju da se postojeći podaci reorganizuju za indeks.[7]

U sledećem primeru SalesOrderDetailID je clustered indeks. Imamo i sledeći primer upita za izdvajanje željenih podataka a to su vrednosti kolona CarrierTrackingNumber i UnitPrice iz tabele SalesData kod onih redova tabele koja u koloni SalesOrderDetailId imaju vrednost 6:

```
SELECT CarrierTrackingNumber, UnitPrice
FROM SalesData
WHERE SalesOrderDetailID = 6
```

Slika 4: Upit u kom se koristi clustered indeks



SalesOrderID	SalesOrderDetailID	CarrierTrackingNumber	OrderQty	ProductID	SpecialOfferID	UnitPrice
43659	1	4911-403C-98	1	776	1	2024.994
43659	2	4911-403C-98	3	777	1	2024.994
43659	3	4911-403C-98	1	778	1	2024.994
43659	4	4911-403C-98	1	771	1	2039.994
43659	5	4911-403C-98	1	772	1	2039.994
43659	6	4911-403C-98	2	773	1	2039.994
43659	7	[illegible]	[illegible]	774	1	2039.994
43659	8	[illegible]	[illegible]	[illegible]14	1	28.8404
43659	9	[illegible]	[illegible]	[illegible]16	1	28.8404
43659	10	[illegible]	[illegible]	[illegible]09	1	5.70
43659	11	[illegible]403C-98	2	712	1	5.1865
43659	12	4911-403C-98	4	711	1	20.1865
43660	13	6431-4D57-83	1	762	1	419.4589
43660	14	6431-4D57-83	1	758	1	874.794
43661	15	4E0A-4F89-AE	1	745	1	809.76

Slika 5: Primer clustered indeksa nad kolonom SalesOrderDetailID

2.5 Non-clustered indeksi

Non-clustered indeksi su smešteni na jednoj lokaciji, dok su podaci iz tabele nad kojom su indeksi kreirani smešteni na sasvim drugoj lokaciji. Indeks sadrži pointere na lokaciju na kojoj se nalaze podaci iz tabele. Jedna tabela može imati više non-clustered indeksa s tim da je indeks koji je non-clustered smešten na različitim lokacijama.[10]

Primera radi, knjiga može imati jedan ili više indeksa, na početku onaj koji određuje sadržaj knjige kao i onaj na kraju koji prikazuje pojmove koji se nalaze u knjizi uređene po azbuci.[10]

Non-clustered indeksi su definisani na poljima tabele koja nisu uređena. Ovaj tip indeksa ne uređuje redove tabele kao što je to slučaj sa clustered indeksima. Ovaj tip indeksiranja omogućava da se poboljšaju performanse upita koji koriste ključeve koji nisu označeni kao

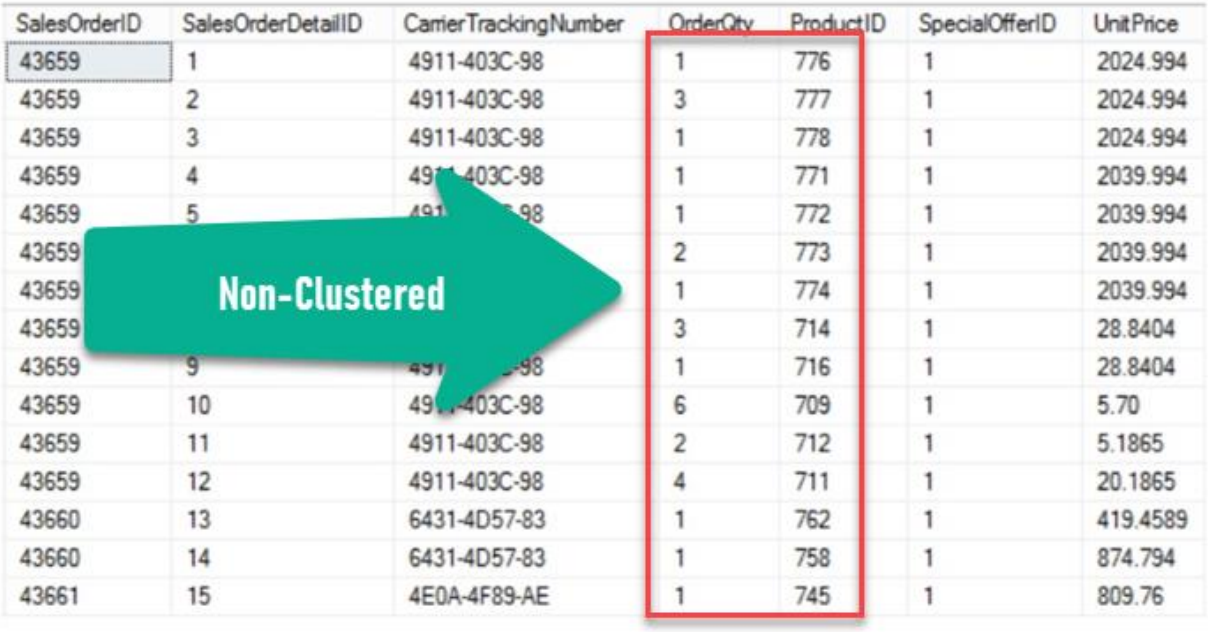
primarni ključevi. Non-clustered indeks omogućava da se doda jedinstveni (unique) ključ u tabelu.[10]

Ako imamo upit koji koristi kolonu nad kojojm je non-clustered indeks kreiran, baza podataka će u toku izvršenja upita prvo otići na indeks date tabele kako bi našao adresu reda koji odgovara upitu, a nakon toga će otići na tu adresu i iz tabele pokupiti potrebne vrednosti iz tog reda.

Non-clustered indeks je u sledećem primeru kreiran nad OrderQty i ProductID:[10]

```
CREATE INDEX myIndex ON
SalesData (ProductID, OrderQty)
```

Slika 6: Kreiranje non-clustered indeksa



SalesOrderID	SalesOrderDetailID	CarrierTrackingNumber	OrderQty	ProductID	SpecialOfferID	UnitPrice
43659	1	4911-403C-98	1	776	1	2024.994
43659	2	4911-403C-98	3	777	1	2024.994
43659	3	4911-403C-98	1	778	1	2024.994
43659	4	4911-403C-98	1	771	1	2039.994
43659	5	4911-403C-98	1	772	1	2039.994
43659			2	773	1	2039.994
43659			1	774	1	2039.994
43659			3	714	1	28.8404
43659	9	4911-403C-98	1	716	1	28.8404
43659	10	4911-403C-98	6	709	1	5.70
43659	11	4911-403C-98	2	712	1	5.1865
43659	12	4911-403C-98	4	711	1	20.1865
43660	13	6431-4D57-83	1	762	1	419.4589
43660	14	6431-4D57-83	1	758	1	874.794
43661	15	4E0A-4F89-AE	1	745	1	809.76

Slika 7: Non-clustered indeks

Sledeći upit se izvršava brže nego prethodni na clustered indeksu.[10]

```
SELECT Product ID, OrderQty
FROM SalesData
WHERE ProductID = 714
```

43659	7	4911-403C-98	1	774	1	2039.994
43659	8	4911-403C-98	3	714	1	28.8404
43659	9	4911-403C-98	1	716	1	28.8404

Slika 8: Upit koji koristi non-clustered indeks

3 Indeksne strukture

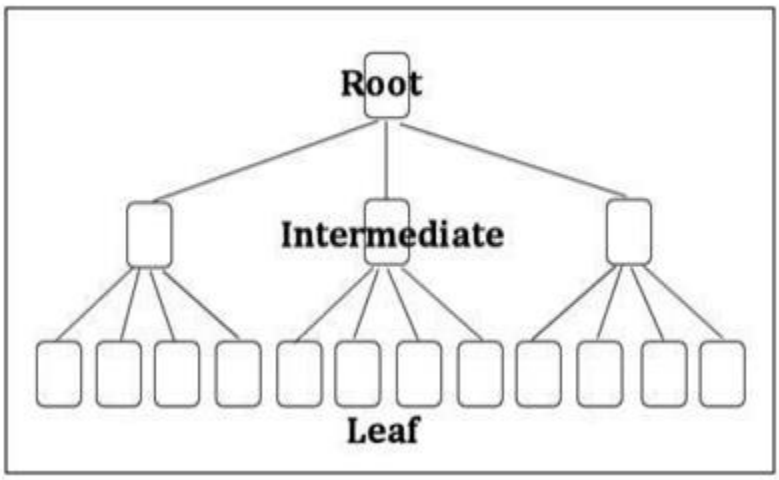
Struktura indeksa može biti predstavljena R-stablima, B-stablima kao i Hash tabelama.

3.1 Indeksi predstavljeni B-stablima

Idealna struktura indeksa koji su predstavljeni B-stablom izgleda kao na slici 6. Generalno indeks ima stranicu koja se nalazi u korenu stable koja ima 0 ili više središnjih slojeva na koje se nadovezuje nivo koji sadrži listove. [2]

Generalna ideja ovakve organizacije indeksa je ta da su vrednosti poređane po redu i svaka stranica koja predstavlja list stabla je na istoj distanci od korena stabla. [3]

Indeksi struktuirani u B-stablo povećavaju brzinu pristupa podacima zato što mehanizam za skladištenje ne mora da pretražuje celu tabelu kako bi našao željene podatke. Umesto toga, pretraga se počinje iz korena stabla. Slotovi u korenu stabla sadrže pointere na čvorove potomke, i mehanizam za skladištenje prati ove pointere. Pravi tj. traženi pointer se nalazi tako što se posmatraju vrednosti na stranicama koje se nalaze na čvorovima, koji definišu gornje i donje granice vrednosti na čvorovima potomcima. Konačno, mehanizam za skladištenje ili određuje da željena vrednost uopšte ne postoji u tabeli ili pak uspešno nalazi odgovarajuću stranicu koja je predstavljena listom stabla. Stranice koje se nalaze na listovima su specijalne stranice, zato što one imaju pointere na indeksirane podatke umesto pointera na druge stranice.



Slika 9: Indeksna struktura u vidu B-stabla (koren, središnji sloj i listovi stabla)

Sledećim primerom pokazaćemo kako upit može dobiti određene benefite korišćenjem indeksa koji su strutkuirani kao B-stablo. [3]

Recimo da imamo sledeću tabelu:

CREATE TABLE People (
last_name varchar(50) not null,
first_name varchar(50) not null,
dob date not null,
gender enum(‘m’, ‘f ’) not null,
key(last_name, first_name, dob)
);

Za navedenu tabelu, indeks bi saržao vrednosti iz kolona *last_name*, *first_name* i *dob* i to iz svakog reda tabele. Indeks će sortirati ove vrednosti na osnovu ovih kolona. Tako da, ako ima ljudi sa istim imenom ali različitim datumom rođenja, oni će biti sortirani po datumu rođenja. [3]

Kada smo naveli kako će i na koji način biti sortirani redovi tabele tj. njihovi indeksi navešćemo gde se to mogu koristiti indeksi. Oni se mogu koristiti za poređenje kolona u izrazima koji koriste =, >, >=, <, <= ili BETWEEN operatori. Indeksi se takođe mogu koristiti za LIKE poređenja ali ako je argument prosleđen za ovaj operator je konstantni string koji ne počinje sa wildcard-om. Na primer, sledeći SELECT izraz koristi indekse:[4]

```
1  SELECT * FROM tbl_name WHERE key_col LIKE 'Patrick%';
2  SELECT * FROM tbl_name WHERE key_col LIKE 'Pat%_ck%';
```

Slika 10: Primer upita

U prvom izrazu, jedino redovi sa 'Patrick' <= ***key_col*** < 'Patricl' su uzeti u obzir. U drugom izrazu, jedino redovi sa 'Pat' <= ***key_col*** < 'Pau' su uzeti u obzir.[4]

Sledeći SELECT izraz ne koristi indekse:[4]

```
SELECT * FROM tbl_name WHERE key_col LIKE '%Patrick%';
SELECT * FROM tbl_name WHERE key_col LIKE other_col;
```

Slika 11: Primer upita

U prvom izrazu LIKE vrednost je počinje wildcard karakterom. U drugom izrazu vrednost za LIKE nije konstanta.

Ako se koristi ... LIKE '%***string***%' i string nije veći od tri karaktera, MySQL koristi Turbo Bajer-Murov algoritam kako bi se inicijalizovao obrazac za string i onda koristi obrazac kako bi se pretraga izvršila mnogo brže.

Pretraga korišćenjem col_name IS NULL upotrebljava indekse ako je col_name indeksirana kolona.[4]

Svaki indeks koji nije u opsegu svih AND nivoa u WHERE klauzuli ne koristi se za optimizaciju upita. Drugim rečima, da bismo bili u mogućnosti da koristimo indeks, prefiks indeksa mora biti korišćen u svakoj AND grupi.

Sledeće WHERE klauzule koriste indekse:

```
... WHERE index_part1=1 AND index_part2=2 AND other_column=3

    /* index = 1 OR index = 2 */
... WHERE index=1 OR A=10 AND index=2

    /* optimized like "index_part1='hello'" */
... WHERE index_part1='hello' AND index_part3=5

    /* Can use index on index1 but not on index2 or index3 */
... WHERE index1=1 AND index2=2 OR index1=3 AND index3=3;
```

Slika 12: Where klauzule koje koriste indekse

Sledeće WHERE klauzule ne koriste indekse:

```
    /* index_part1 is not used */
... WHERE index_part2=1 AND index_part3=2

    /*  Index is not used in both parts of the WHERE clause  */
... WHERE index=1 OR A=10

    /* No index spans all rows  */
... WHERE index_part1=1 OR index_part2=10
```

Slika 13: Where klauzule koje koriste indekse

Ponekad MySQL ne koriste indeks, iako je on dostupan. Jedina okolnost pod kojom se to događa je kada optimzator procenjuje da bi korišćenjem indeksa trebalo da MySQL pristupi veoma velikom procentu redova u tabeli. U ovom slučaju pretraga cele tabele će biti puno brže jer zahteva manje traženja. Međutim, ako takav upit koristi LIMIT za pristupanje samo nekim redovima, MySQL koristi indeks svakako, zato što može mnogo brže naći nekoliko redova koje će vratiti kao rezultat.

3.2 Karakteristike Hash indeksa

Hash indeksi imaju neke različite karakteristike od indeksa predstavljenjih B-stablom:

Ovakva vrsta indeksa koristi se jedino za poređenja jednakosti koja koriste = ili <=> operatore (ali su veoma brzi). Ne koriste se za operatore poređenja kao što je < koji nalazi raspon vrednosti. Sistemi koji se oslanja na ovaj tip jednovrednosnih pretraga poznati su kao “ključ-vrednost skladišta”. Ako se želi korišćenje MySQL-a za ovakve vrste aplikacija, potrebno je koristiti hash indekse gde god je to moguće.[3]

Optimizator ne može koristiti hash indekse kako bi ubrzao ORDER BY operacije. Ovaj tip indeksa ne može biti korišćen za traženje sledećeg unosa po redu.[3]

MySQL ne može aprokismalno odrediti koliko redova postoji između dve vrednosti (ovo je iskorišćeno od strane optimizatora opsega, kako bi odlučio koji indeks da koristi). Ovo može uticati na pojedine upite ako se izmeni MyISAM ili InnoDB tabela u MEMORY tabelu koja je indeksirana hash indeksom.[3]

Jedino celoviti ključevi mogu biti korišćeni za pretragu redova. (Sa indeksima koji su u strukturi B-stabla, jedini najleviji prefiks ključa može se koristiti da bi se pronašli redovi)[3]

3.3 Indeksi predstavljeni R-stablima

Indeksi predstavljeni R-stablima se koriste za prostorne ili N-dimenizonalne podatke. Vrlo su popularni kod aplikacija koje sadrže mapiranje i bave se geografijom ali takođe su se dobro pokazali i u drugim situacijama u kojima su pojedini rekordi pamte na osnovu dve ose ili dimenzija kao što su dužina i širina ili visina i težina itd.[13]

Ključna ideja ovakve strukture podataka je da se grupišu objekti koji su blizu i da se predstave tj obuhvate minimalnim pravougaonikom u sledećem višem nivou stabla. Obzirom da svi objekti leže u ovom ograničavajućem pravougaoniku, upit koji ne preseca granični pravougaonik takođe ne može obuhvatiti nijedan od sadržanih objekata.[13]

4 Tipovi indeksa

U ovom poglavlju pokazaćemo koji sve tipovi indeksa se mogu dodati u MySQL tabele kao i u kojim situacijama korišćenje kog tipa indeksa treba preferirati.

4.1 Pet tipova indeksa

Prilikom kreiranja i dodavanja indeksa u tabelu, može se kreirati jedan od sledećih tipova indeksa.

4.1.1 Unique indeks

Unique indeks je takav indeks u kome vrednosti svih kolona moraju biti jedinstvene. U unique indeksu koji sadrži samo jednu kolonu ne može biti ponovljenih vrednosti u kolonama koje su indeksirane tj. ne može biti duplikata. U unique indeksu koji sadrži više kolona vrednosti mogu biti duplikati ali samo u okviru jedne kolone, ali kombinacije vrednosti u svim kolonama koje čine indeks moraju biti jedinstvene. Unique indeksi koriste se kako bi se izbeglo dopliciranje vrednosti i ovakav indeks se uglavnom kreira nakon što je tabela kreirana.[7]

4.1.2 Primarni ključ

Primarni ključ predstavlja unique indeks u kome vrednosti ne smeju biti NULL. Svaki red mora imati vrednost za zadatu kolonu ili kombinaciju kolona. Zbog toga se obično primarni ključ definiše na najmanjem mogućem broju kolona, i u većini slučajeva primarni ključ će biti

postavljen na samo jednu kolonu. Svakako, kada se jednom postave vrednosti u koloni koja označava primarni ključ, te vrednosti se više ne mogu menjati. Primarni ključ se definiše pnajčešće prilikom kreiranja tabele.[7]

4.1.3 Regularni (normalni) indeks

Ovaj indeks dozviljava da njegove vrednosti ne budu jedinstvene a takođe mogu biti NULL. Ovaj indeks dodaje se, kako bi se omogućilo da se u bazi podataka brže pronalaze željeni podaci. [7]

4.1.4 Fulltext indeks

Ova vrsta indeksa primenjuje se za pretrage celih tekstova. Ponekad kada je potrebno pronaći deo teksta koji sadrži određenu reči ili grupu reči, ili ako se želi pronaći određeni podstring u velikom tekstualnom bloku.

Umesto da se cela vrednost indeksira, fulltext indeks će indeksirati pojedinačne reči u svakom tekstualnom bloku. Ovo omogućava brže nalaženje specifičnih reči i fraza u celom tekstu.[7]

4.1.5 Opadajući indeks

Ova vrsta indeksa dostupna je u verziji 8+ MySQL i predstavlja regularni indeks koji je uređen u opadajući redosled. Od pomoći je u situacijama kad je potrebno izvršiti upit za najskorije dodate podatke, na primer ako želimo prikazati pet najskorijih postova ili deset najnovijih komentara na svim svojim opostovima.[7]

5 Indeksi i tipovi tabela u MySQL

U prethodnim poglavljima govorili smo o tipovima indeksa, terminologiji koja se koristi u vezi indeksa, a u ovom poglavlju govorićemo o tome kako su i koji indeksi implementirani u različitim mehanizmima za skladištenje u MySQL-u. Svaki od mehanizama implementira neke od indeksnih struktura koje smo spomenuli. Takođe svaki od mehanizama pruža različite mogućnosti koje se tiču optimizacije.[13]

5.1 MyISAM tabele

Podrazumevani tip tabela za skladištenje podataka u MySQL-u je upravo MyISAM. Ovakva vrsta tabela omogućavaju indeksnu strukturu podataka predstavljenu u vidu B-stabla. Od verzije 4.1.0 omogućena je i struktura indeksa predstavljena u vidu R-stabla za prostorni tip podataka. Pored benefita koje pruža dobra implementacija indeksa u vidu B-stabla, MyISAM dodaje dve vrlo važne ali relativno nepoznate prednosti a to su *prefix compression* i *packed keys.[13]*

Prefix compression koristi se kako bi se izbegli učestali prefiksi u ključevima koji su predstavljeni stringom. [13]

Packed keys je najbolje objasniti tako što ćemo reći da on predstavlja isto što i prefix compression samo za ključeve koji su integer tipa. Naime obzirom da su integer ključevi u memoriji zapamćeni tako što se nijhovi viši bajtovi pamte prvi, najbolje je za veću grupu ključeva deliti učestale prefikse, zato što se najviši bitovi broja najređe menjaju.[13]

MySQL čuva indekse za tabele kada je u pitanju korišćenje MyISAM tabela u .MYI fajlu tabele.[13]

5.2 InnoDB tabele

Kod InnoDB tabela omogućeno je kreiranje indeksa koji su predstavljeni B-stablima. Za razliku od MyISAM nije omogućena prefix compression niti packed keys. Pored toga InnoDB takođe zahteva primarni ključ za svaku tabelu. Ako se ne dodeli primarni ključ MySQL će sam dodeliti tabeli 64-bitnu vrednost.[13]

Indeksi se čuvaju u InnoDB tabelarnom prostoru baš kao i podaci i rečnik podataka (definicije tabela itd.). Takođe InnoDB koristi clustered indekse. U skladu s tim vrednost primarnog ključa direktno utiče na fizičku lokaciju reda u tabeli kao i na lokaciju njegovog indeksnog čvora. Zbog ovoga pretrage koje se baziraju na primarnom ključu u InnoDB su veoma brze. Jednom kada se indeksni čvor pronađe, relevantni rekordi tabele će biti keširani u InnoDB baferu.[13]

6 Rad sa indeksima

U ovom poglavlju opisaćemo načine za kreiranje, uklanjanje i preimenovanje indeksa.

6.1 Kreiranje indeksa

Kreiranje indeksa može se obaviti na dva načina.

Prvi način predstavlja kreiranje indeksa za tabelu prilikom kreiranja same tabele tj. naredbom CREATE TABLE.[6] Sledeći primer prikazuje kreiranje nove tabele sa indeksom:

```
CREATE TABLE table_name
(
  column1 datatype [ NULL | NOT NULL ],
  column2 datatype [ NULL | NOT NULL ],
  ...
  column_n datatype [ NULL | NOT NULL ],

  INDEX index_name [ USING BTREE | HASH ]
    (index_col1 [(length)] [ASC | DESC],
     index_col2 [(length)] [ASC | DESC],
     ...
     index_col_n [(length)] [ASC | DESC])
);
```

Slika 14: Kreiranje tabele i indeksa u jednoj naredbi

Drugi način kreiranja indeksa može se obaviti nakon kreiranja same tabele i to naredbom CREATE INDEX kojom se može kreirati indeks nad jednom ili više kolona. Sintaksa za kreiranje indeksa na ovaj način je:[6]

```
CREATE [UNIQUE | FULLTEXT | SPATIAL] INDEX index_name
  [ USING BTREE | HASH ]
  ON table_name
    (index_col1 [(length)] [ASC | DESC],
     index_col2 [(length)] [ASC | DESC],
     ...
     index_col_n [(length)] [ASC | DESC]);
```

Slika 15: Kreiranje indeksa nakon kreiranja tabele

UNIQUE
: Opcionalno. *UNIQUE* modifikator određuje da kombinacija vrednosti iz datih kolona koje čine indeks mora biti jedinstvena.[6]

FULLTEXT
: Opcionalno. *FULLTEXT* modifikator indeksira celu kolonu bez dozvole korišćenja prefiksa. InnoDB i MyISAM imaju ovu opciju.[6]

SPATIAL
: Opcionalno. *SPATIAL* modifikator indeksira celu kolonu i ne dozvoljava da kolona koja je indeksirana sadrži NULL vrednosti. InnoDB (od vrezije MySQL 5.7) i MyISAM tabele imaju ovu opciju.[6]

index_name
: Ime dodeljeno indeksu.[6]

table_name
: Ime tabele za koju se kreira indeks.[6]

index_col1, index_col2, ... index_col_n
: Kolone nad kojima se kreira indeks.[6]

length
: Opcionalno. Ako je specificiran, jedino prefiks za datu kolonu je indeksiran, a ne cela kolona. Za nebinarne kolone koje sadrže stringove, ova vrednost je zadat broj karaktera kolone za indeksiranje. Za binarne kolone koje sadrže stringove, ova vrednost predstavlja broj bajtova kolone za indeksiranje.[6]

ASC
: Opcionalno. Indeks je sortiran u rastući redosled na osnovu date kolone.[6]

DESC
: Opcionalno. Indeks je sortiran u opadajući redosled na osnovu date kolone.[6]

Podrazumevano, MySQL kreira indeks koji je struktuiran u B-stablo ako se ne specificira tip indeksa koji se kreira. Sledeća tabela pokazuje moguće tipove indeksa na osnovu mehanizma za skladištenje tabele:[5]

Storage Engine	Allowed Index Types
InnoDB	BTREE
MyISAM	BTREE
MEMORY/HEAP	HASH, BTREE

Slika 16: Mogući tipovi indeksa za svaku od mehanizma za skladištenjes

6.1.1 Primer kreiranja indeksa

Sledeći upit pronalazi zaposlene čiji opis pozicije je Sales rep:

```
1  SELECT
2      employeeNumber,
3      lastName,
4      firstName
5  FROM
6      employees
7  WHERE
8      jobTitle = 'Sales Rep';
```

Slika 17: Primer upita

Ovim upitom dobijamo sledeći rezultat:

employeeNumber	lastName	firstName
1165	Jennings	Leslie
1166	Thompson	Leslie
1188	Firrelli	Julie
1216	Patterson	Steve
1286	Tseng	Foon Yue
1323	Vanauf	George
1337	Bondur	Loui
1370	Hernandez	Gerard
1401	Castillo	Pamela
1501	Bott	Larry

Slika 18: Dobijeni rezultati nakon upita

Kako bismo videli kako MySQL interno izvodi prethodni upit dodajemo EXPLAIN klauzulu na početak SELECT upita:

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	employees	NULL	ALL	NULL	NULL	NULL	NULL	23	10.00	Using where

Slika 19: Tabela sa prikazom objašnjenja upita

Kao što možemo primetiti, MySQL je morao da skenira celu tabelu koja se sastoji od 23 reda kako bi našao zaposlene koji odgovaraju uslovima upisa.

Sada ćemo kreirati indeks sa kolonom jobTitle koristeći CREATE INDEX izraz:

```
1 CREATE INDEX jobTitle ON employees(jobTitle);
```

Slika 20: Kreiranje indeksa

Izvršićemo upit ponovo:

```
1 EXPLAIN SELECT
2     employeeNumber,
3     lastName,
4     firstName
5 FROM
6     employees
7 WHERE
8     jobTitle = 'Sales Rep';
```

Slika 21: Primer upita

Imaćemo sledeći rezultat u tabeli objašnjenja upita:

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	employees	NULL	ref	jobTitle	jobTitle	52	const	17	100.00	NULL

Slika 22: Tabela sa prikazom stanja upita

Kao što se može primetiti, MySQL je upravo locirao 17 redova iz jobTitle indeksa kao što je naznačeno u koloni ključeva bez skeniranja cele tabele.

Kako bi se prikazali svi kreirani indeksi nad jednom tabelom koristi se naredba SHOW INDEXES, kao u sledećem primeru:

```
1 SHOW INDEXES FROM employees;
```

Slika 23: Naredba za prikaz indeksa tabele

Dobijamo sledeći rezultat:

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_commer	Visible
employees	0	PRIMARY	1	employeeNumber	A	23	NULL	NULL		BTREE			YES
employees	1	reportsTo	1	reportsTo	A	7	NULL	NULL	YES	BTREE			YES
employees	1	officeCode	1	officeCode	A	7	NULL	NULL		BTREE			YES
employees	1	jobtitle	1	jobTitle	A	7	NULL	NULL		BTREE			YES

Slika 24: Rezultat prethodne naredbe

6.2 Brisanje indeksa

Kako bi se uklonio postojeći indeks iz tabele, koristi se DROP INDEX naredba kao što je u sledećem primeru prikazano:

```
1 DROP INDEX index_name ON table_name
2 [algorithm_option | lock_option];
```

Slika 25: Naredba za brisanje indeksa

U ovoj sintaksi:

- Prvo, specificira se ime indeksa koji se želi ukloniti nakon DROP INDEX ključnih reči (index_name)

- Drugo, specificira se ime tabele kojoj indeks pripada (table_name)

Algoritam

Algorithm_option dopušta da se specificira algoritam koji se koristi za uklanjanje indeksa. Sledeći primer prikazuje sintaksu algorithm_option klauzule:

```
1 ALGORITHM [=] {DEFAULT|INPLACE|COPY}
```

Slika 26: Algorithm naredba

Za uklanjanje indeksa podržani su sledeći algoritmi:[5]

- COPY: Tabela se kopira u novu tabelu red po red, DROP INDEX se zatim primenjuje na kopiju originalne tabele. Manipulacija nad trenutnim podacima sa INSERT ili UPDATE naredbama nisu dozvoljene.
- INPLACE: Tabela je prepravljena na licu mesta umesto kopiranja u novu tabelu. MySQL izdaje ekskluzivno zaključavanje metapodataka u tabeli tokom prepravljanja i izvršenja faza operacije uklanjanja indeksa. Ovaj algoritam omogućava konkurentne inaredbe za manipulaciju podacima.

Naglasićemo da je klauzula ALGORITHM opcionalna, Ako se ova klauzula izostavi, MySQL koristi INPLACE. U slučaju da INPLACE nije podržan, MySQL koristi COPY.[5]

Korišćenje DEFAULT ima isti efekat kao izostavljanje ALGORITHM klauzule.[5]

6.3 Preimenovanje indeksa

Postoji mogućnost preimenovanja indeksa u MySQL. U zavisnosti od vrezije MySQL-a postoje tri različite sintakse za preimenovanje indeksa.

Sintaksa za preimenovanje indeksa korišćenjem ALTER TABLE naredbe u MySQL 5.6 i ranijim verzijama je sledeća:

```
ALTER TABLE table_name
  DROP INDEX index_name,
  ADD INDEX new_index_name [ USING BTREE | HASH ]
    (index_col1 [(length)] [ASC | DESC],
     index_col2 [(length)] [ASC | DESC],
     ...
     index_col_n [(length)] [ASC | DESC]);
```

Slika 27: Naredba za premenovanje indeksa

Ili u MySQL 5.7 i novijim verzijama je:

```
ALTER TABLE table_name
  RENAME INDEX index_name TO new_index_name;
```

Slika 28: Naredba za preimenovanje indeksa u MySQL 5.7 i novijim verzijama

table_name
: Ime tabele na kojoj je indeks kreiran.

index_name
: Ime indeksa koji želimo da preimenujemo.

new_index_name
: Novo ime za indeks.

6.3.1 Primer preimenovanja indeksa

Prikazaćemo primer preimenovanja indeksa u MySQL-u. U starijim verzijama MySQL-a, koristi se ALTER TABLE naredba kako bi se prvo uklonio stari indeks a onda ponovo kreirao novi indeks. Primer (MySQL 5.6 i ranije verzije):

```
ALTER TABLE contacts
  DROP INDEX contacts_idx,
  ADD INDEX contacts_new_index (last_name, first_name);
```

Slika 29: Primer naredbe preimenovanja indeksa

U ovom primeru izvršeno je preimenovanje indeksa pod nazivom *contacts_idx* u ime *contacts_new_index.* Ovo je urađeno tako što je uklonjen stari indeks i dodat je novi indeks.

Počevši od MySQL verzije 5.7, moguće je korišćenje ALTER TABLE naredbe sa RENAME INDEX klauzulom kako bi se preimenovao indeks. Primer (MySQL 5.7 i novije verzije):

```
ALTER TABLE contacts
  RENAME INDEX contacts_idx TO contacts_new_index;
```

Slika 30: Primer preimenovanja indeksa

Ako niste sigurni u kojoj verziji MySQL-a trenutno radite, najsigurnije je da koristite prvu navedenu sintaksu za preimenovanje indeksa.

7 Zaključak

Indeksiranje je najbitnija tehnika koju posedujemo za ubrzavanje izvršenja upita. Druge tehnike su takođe dostupne, ali tehnika koja pravi najveću razliku u brzini izvršenja je ipak tehnika koja koristi indeksiranje. Na mailing listi MySQL-a ljudi najčešće pitaju za pomoć kako bi se njihovi upiti izvršavali brže. U iznenađujućem broju takvih slučajeva ne postoje indeksi za tabele koje pomenuti ljudi koriste, a dodavanje indeksa često odmah rešava njihove probleme sa ubrzavanjem upita. Ipak ne funkcioniše uvek sve tako, zato što ipak optimizacija upita nije uvek tako jednostavna. Kako god bilo, ako se ne koriste indeksi u mnogim slučajevima samo se gubi vreme pokušavajući da se performanse poboljšaju drugim tehnikama. Preporuka je da se uvek prvo koriste indeksi kako bi se njima omogućilo najveće povećanje performansi, a nakon toga se mogu uključiti i druge tehnike koje mogu dodatno doprineti performansama.

8 Literatura

[1] Web-sajt: https://dev.mysql.com/doc/refman/8.0/en/mysql-indexes.html

[2] Web-sajt: https://arxiv.org/ftp/arxiv/papers/1903/1903.08334.pdf

[3] Web-sajt: https://guptavikas.wordpress.com/2012/12/17/b-tree-index-in-mysql/

[4] Web-sajt: https://dev.mysql.com/doc/refman/8.0/en/index-btree-hash.html#btree-index-characteristics

[5] Web-sajt: https://www.mysqltutorial.org/mysql-index/mysql-create-index/

[6] Web-sajt: https://www.techonthenet.com/mysql/indexes.php

[7] Web-sajt: https://vanseodesign.com/web-design/the-types-of-indexes-you-can-add-to-mysql-tables/

[8] Web-sajt: https://www.mysqltutorial.org/mysql-index/mysql-clustered-index/

[9] Web-sajt: https://dev.mysql.com/doc/refman/5.7/en/innodb-index-types.html

[10] Web-sajt: https://www.guru99.com/clustered-vs-non-clustered-index.html

[11] Web-sajt: https://www.sqlshack.com/what-is-the-difference-between-clustered-and-non-clustered-indexes-in-sql-server/

[12] Web-sajt: https://en.wikipedia.org/wiki/Database_index

[13] Web-sajt: https://www.oreilly.com/library/view/high-performance-mysql/0596003064/ch04.html