LINKgroup

Distance Learning System

MySql

Indeksi, pogledi, uskladištene rutine i transakcije

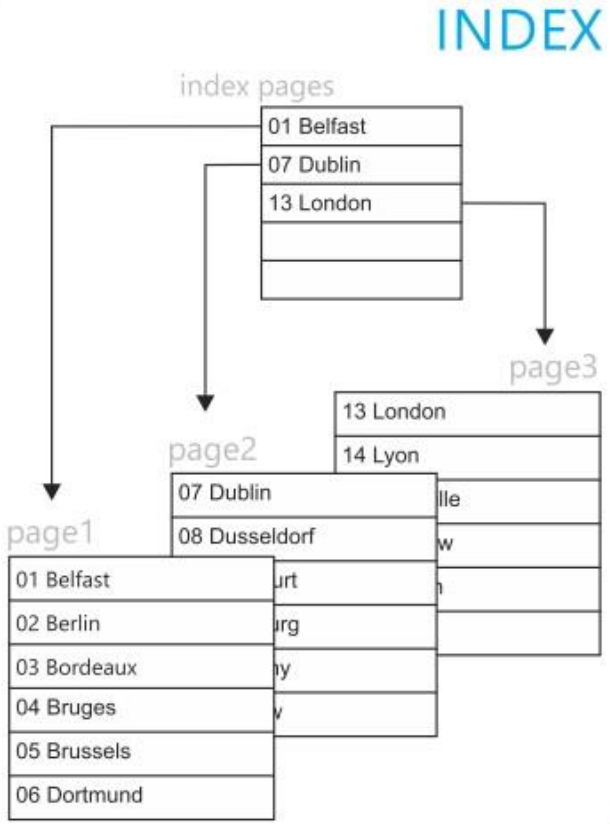
Indeksi

- U rukovanju bazama podataka, ključna stvar je brzo i precizno dobavljanje podataka. Kada su količine podataka u tabelama male, sistem će, u svakom slučaju, funkcionisati brzo. Ali, kada baza počne da sadrži nešto veće količine podataka, brzina postaje jedan od osnovnih problema. Međutim, nije brzina dostavljanja podataka ta koja proces čini sporim. Reč je o samom pronalaženju podataka. Kada se podatak jednom locira, brzina njegovog dostavljanja je irelevantna

1	Paris
2	Bruges
3	London
4	Moscow
5	Krakow

Mehanizam indeksa

- Većina podataka u MySQL bazi pamti na stranama. Veličina jedne strane je ograničena memorijski (obično su 8k ili 16k), ali ne može se tačno znati koliko u tu stranu može stati redova jer to zavisi od raznih faktora. Pre svega, zavisi od količine podataka u redovima.
- Ovako izgledaju strane za tabelu sa gradovima



Klasterovani i neklasterovani indeks

- Pretraga indeksa uvek funkcioniše na isti način, ali se njena finalizacija može izvršiti na dva načina. Ova dva načina čine da razlikujemo i dve vrste indeksa:
  - **klasterovani**
  - **neklasterovani (sekundarni)**
- Često se, sa punim pravom, ova dva indeksa porede sa knjigom i telefonskim imenikom. Ukoliko tražimo neku informaciju iz knjige, prvo ćemo pretražiti sadržaj i u njemu naći stranu za ono što nas zanima. Zatim ćemo prelistati knjigu i brzo doći do podatka, jer znamo stranu. Ako tražimo neki podatak u telefonskom imeniku, naći ćemo slovo koje nas zanima, a zatim direktno pristupiti strani na kojoj se nalaze svi podaci koji počinju tim slovom.
- Klasterovani indeks, na svom najnižem nivou (Leaf level) sadrži zapravo same podatke, dok neklasterovani indeks sadrži reference na prave podatke.
- Zato su neklasterovani indeksi nešto sporiji od klasterovanog

Formiranje Klasterovanog i neklasterovanog indeksa

- Svaki put kada kreiramo primarni ključ, automatski se kreira i klasterovani indeks. Ovaj indeks može biti samo jedan na nivou tabele
- Neklasterovane indekse kreiramo eksplicitno, i može ih biti više po jednoj tabeli

Strategija indeksiranja (klasterovani indeks)

- Kada napravimo jednu tabelu i dodelimo joj primarni ključ, MySQL server automatski, po toj koloni, kreira klasterovani indeks, pa je samo bitno da dobro isplaniramo šta će biti primarni ključ naše tabele, dok na sam klasterovani indeks i nećemo imati preveliki uticaj

```
create table mytable
(
id int primary key auto_increment,
name varchar(256)
);
```

```
show index from mytable
```

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type
mytable	0	PRIMARY	1	id	A	0	NULL	NULL		BTREE

Kreiranje neklasterovanog indeksa

- Neklasterovani indeks je malo pristupačniji i njega možemo postaviti gde god hoćemo.
- To ne znači da je pametno staviti ga na sve kolone tabele
- Indeks je, još jedna velika količina podataka koja će se naći na našem serveru. Takođe, to je i vlika količina podataka koje naš sistem treba da obrađuje.
- Indeksi su u stanju da znatno ubrzaju pretragu, ali, njima se takođe pristupa i prilikom drugih intervencija na tabelama.
- Što više indeksiranih kolona u tabeli, utoliko više usporenja uzrokovanih obradom indeksa prilikom rukovanja podacima te tabele.

```
ALTER TABLE mytable ADD INDEX custom_nonclustered_index (name ASC)
```

Full text index

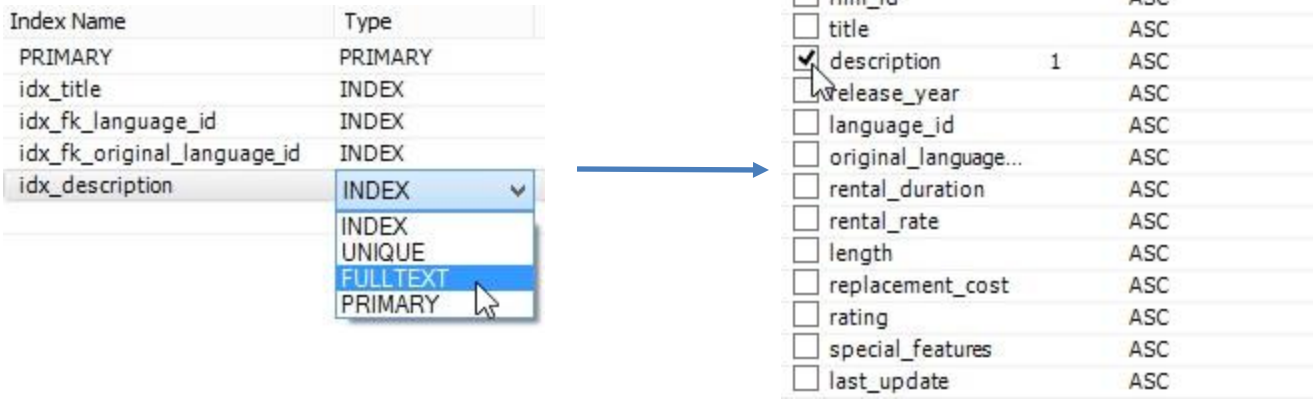
- Standardni indeks sa kojim smo se upoznali u dosadašnjem toku lekcije, kod tekstualnih kolona, sposoban je da prilikom pretrage uzme u razmatranje samo početne karaktere unosa. U situacijama kada u nekoj tekstualnoj koloni imamo zabeležen tekst koji se sastoji od više reči, običan indeks je praktično beskoristan.
- U takvim situacijama na scenu stupa Full-text Index.
- Full-text Index omogućava brzu pretragu kroz velike količine teksta. Ovo je dobar, funkcionalan, jednostavan i brz sistem i, što je najvažnije, nezamenljiv u nekim situacijama.
- Korišćenjem ovog indeksa, prilikom indeksiranja, indekser parsira tekst i preuzima iz njega reči. Svaka reč smešta se u indeks pod određenim brojem. Takođe, polje sa koga je tekst preuzet pamti se pod određenim brojem. Na kraju, indekser pravi relaciju između polja sa tekstom i rečima koje tom polju pripadaju.
- Kada dođe do pretrage, Engine prvo pronađe tražene reči, zatim izlista njihove relacije (u relacijama se pamti i broj pojavljivanja reči u polju), a zatim na osnovu tih relacija sortiranih po broju pojavljivanja reči, dobijamo i sama polja (odnosno, brojeve tih polja)

Kreiranje full text indeksa

- Ponekad ćemo pokušati da izvršimo pretragu na sledeći način:

```
select * from film where description like '%drama%'
```

- Kada kolone imaju mnogo teksta i ima mnogo redova u tabeli, ovo nije dobar koncept. Umesto njega, treba kreirati full text index:



Korišćenje full text indeksa

- Kada je full text indeks postavljen, postaju raspoložive i dodatne komande za pretragu:

```
SELECT * FROM film
WHERE MATCH(description) AGAINST('epic drama')
```

Filtracija rezultata (Search Expressions)

- Pored ovakvog standardnog načina definisanja upita za pretragu kolona korišćenjem Full Text Indexa, postoje i načini na koje možemo da utičemo na samu pretragu

```
SELECT description FROM film
WHERE MATCH(description) AGAINST('+epic +drama' IN BOOLEAN MODE);
```

```
SELECT description FROM film
WHERE MATCH(description) AGAINST('+epic -drama' IN BOOLEAN MODE);
```

Simbol	Značenje
+word	reč se mora pojaviti u rezultatu
-word	reč se ne sme pojaviti u rezultatu
~word	reč se može prijaviti u rezultatu, ali joj je data manja važnost
<word	reči se daje manje značenje prilikom pretrage
>word	reči se daje veće značenje prilikom pretrage
word*	sve reči koje počinju sa word (npr. word, words, wordless)
„word1 word2“	traži se fraza identična onoj navedenoj među navodnicima
()	zagrade se koriste za grupisanje izraza, npr. '+drama +(epic saga) ', tj. traže se svi izrazi koji sadrže fraze drama ili epic saga, ili i jedno i drugo

Pogledi

- Pogledi predstavljaju funkcionalnost SQL-a koja omogućava definisanje specijalne reprezentacije jedne ili više tabela.
- Ovo znači da je korišćenjem pogleda moguće kreirati takozvane virtualne tabele, koje mogu biti kombinacija kolona koje pripadaju različitim tabelama. Nad pogledima je zatim moguće vršiti SELECT upite i, u zavisnosti od definicije pogleda i prava korisnika, izmene podataka INSERT, UPDATE i DELETE naredbama.

Zašto koristimo poglede?

- **Dva najbitnija razloga korišćenja pogleda su sledeći:**
- **Sigurnost**
  - Veoma često može postojati situacija u kojoj želimo da određenim korisnicima baze podataka onemogućimo pun pristup određenoj tabeli ili tabelama.
  - Tipičan primer ove situacije su tabele za smeštanje podataka o zaposlenima. Mi ćemo svakako želeti da omogućimo svim zaposlenima pristup njihovim ličnim podacima, kao što su ime, prezime, broj telefona ili adresa. Ipak, zaposlenima nije dobro omogućiti pristup podacima o visini zarada i sličnim osetljivim podacima.
- **Komfor**
  - Već smo videli da je neophodno često vršiti spajanje podataka više tabela. Na primer, u bazi podataka Sakila, ukoliko bismo želeli da dobijemo kompletan set podataka o mušterijama, morali bismo da izvršimo upit kojim bismo spojili podatke nekoliko tabela. Kako bi se olakšao rad korisnicima baze podataka ili programerima koji će pisati aplikacije koje će komunicirati sa bazom, mi možemo olakšati stvari i kreirati pogled.

Kreiranje pogleda

- Pogledi (View) su objekti u bazi podataka. I zbog toga se, kao i ostali objekti, mogu kreirati DDL naredbom

```
CREATE VIEW myView AS query
```

```
create view uzmifilmove
as
select film.*,film_actor.actor_id from film_actor
join film on film_actor.film_id = film.film_id
```

Ažuriranje izvornih podataka

- Pogledi, osim mogućnosti prikazivanja aktuelnog sadržaja po zadatoj formi, imaju i mogućnost ažuriranja izvora na kojima su formirani, ali pri tom moraju biti ispoštovana neka pravila. Ova pravila se odnose na kreiranje SELECT upita prilikom definisanja pogleda. Ovo znači da se prilikom kreiranja pogleda pri pisanju SELECT upita moraju poštovati sledeća pravila:
- SELECT upit ne sme sadržati ključne reči GROUP BY, DISTINCT, LIMIT, UNION ili HAVING
- Pogledi koji grupišu podatke iz više tabela gotovo nikada se ne mogu koristiti za izmenu konkretnih podataka, tj. SELECT upit mora baratati samo jednom tabelom
- pogled mora sadržati sve kolone određene tabele koje su definisane kao primarni ili strani ključevi

Prava

- Pogledi su usko vezani sa bezbednošću i pravima korisnika. Ovi pojmovi će biti detaljno objašnjeni u sledećem modulu, ali ćemo i sada, ovde, napraviti kratku demonstraciju.
- Prilikom kreiranja pogleda, kreator pogleda mora imati prava nad tabelama nad kojima se pogled kreira. Sa druge strane, korisnik ne mora imati SELECT prava nad tabelama, sve dok ima prava nad pogledom.

Vežba, kreiranje prava nad pogledom

- Kreirati korisnika John i dodeliti mu prava na pogled uzmifilmove

```
CREATE USER 'John'@'localhost' identified by '123'

grant select on uzmifilmove to John
```

- Ulogovati se kao John, a zatim isprobati sledeće upite: