Indeksiranje

SQL Server je deljeni servis kojem istovremeno može da pristupa veliki broj korisnika, pa je od vitalnog značaja da se SQL upiti izvršavaju što je brže i efikasnije moguće. Dobro napisan kod može da pomogne da se upit ubrza, međutim, daleko bitnije je ubrzati ulazno-izlazne (I/O) operacije. Poznato je da se procesorske naredbe izvršavaju za red veličine brže od I/O naredbi. Uzevši u obzir da SQL Server radi sa velikom količinom podataka uskladištenim na disku, postaje jasno da performanse sistema prvenstveno zavise od optimizacije ulaza i izlaza. Najbolji način da se ovo uradi je indeksiranje.

Indeksi su strukture podataka koje ubrzavaju pretraživanje tabela u bazi podataka. Sa indeksima nije neophodno prolaziti kroz svaki red tabele svaki put kada se pristupi bazi podataka.

Podaci na disku se čuvaju organizovani u memorijske blokove. Da bi se pročitao jedan zapis, neophodno je učitati ceo blok. SQL Server podatke organizuje u stranice (*data pages*) od po 8KB. Jedna tabela u bazi podataka se može posmatrati kao skup stranica.

Kada nad tabelom nema kreiranih indeksa, tabela je organizovana kao hip (*heap* - gomila). U hip tabeli, zapisi su razbacani i neuređeni. Kada se dodaju novi podaci, alocira se novi skup stranica koji se dodaje na već postojeći hip. Baza može da sačuva te nove podatke gde god nađe prostora, jer fizički redosled podataka ovde nije bitan. U SQL Serveru, hip organizacija podataka se dobija kada se kreira tabela bez primarnog ključa i bez ikakvih indeksa. Ovakva tabela nije sortirana ni po jednoj koloni.

Kada se zadaje upit nad hip tabelom, *query engine* mora da prođe kroz sve stranice da bi vratio rezultat. Operacija ispitivanja cele tabele se u SQL Serveru zove ***table scan***. Ovakav način izvršavanja upita je prihvatljiv za upite koji treba da vrate celu tabelu kao rezultat:

```
select * from tabela;
```

Da bi se izvršio dati upit, svakako je potrebno proći kroz celu tabelu. Ali za sve ostale vrste upita, prolazak kroz celu tabelu oduzima previše vremena, izvršava se previše I/O operacija, za svaki upit se učitavaju sve stranice koje čine tabelu. Efikasnija alternativa je korišćenje indeksa.

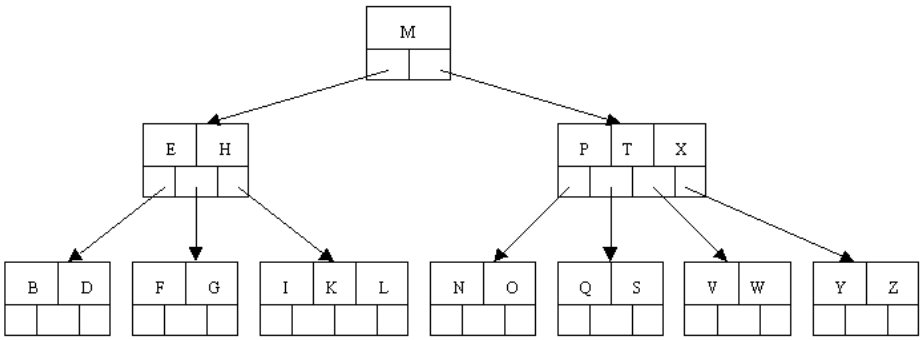
Indeksi u sebi sadrže kopiju dela podataka iz tabele, jednu kolonu ili više njih. Kolone po kojima se vrši indeksiranje čine **ključ pretrage** (*search-key*). Jedan zapis u indeksu izgleda ovako:

ključ pretrage	pokazivač

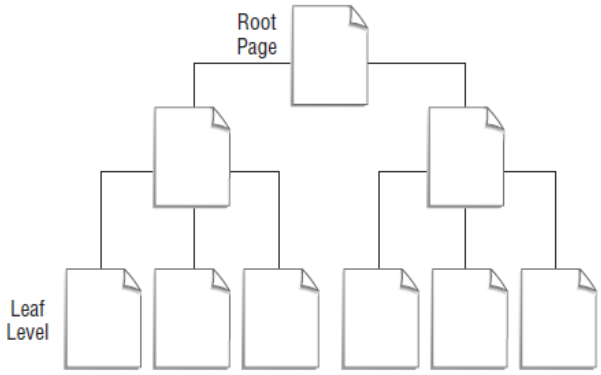
gde pokazivač vodi do zapisa u tabeli koji ima istu vrednost u koloni (kolonama) ključa pretrage.

Indeksi se kreiraju kao balansirana B-stabla (slika 1). U jednom čvoru može biti više podataka, sve grane su iste dubine i iz svakog zapisa se praćenjem levog pokazivača prelazi na manje vrednosti ključa pretrage, a praćenjem desnog pokazivača na veće vrednosti. U indeksu je svaki čvor jedna stranica podataka (slika 2). Podaci u listovima su sortirani po vrednosti ključa pretrage, i pokazivačima su spojeni u dvostruko povezanu listu. Listovi mogu sadržati pokazivače do odgovarajućih podataka u tabeli, a mogu i fizički sadržati same stranice iz tabele, zavisno od vrste indeksa. Cilj indeksa je minimizovanje I/O

operacija tako što se iz razmatranja izbacuju sve nepotrebne stranice tabele, a učitavaju se samo stranice koje sadrže tražene podatke. Indeksi se dele na klasterovane i neklasterovane.



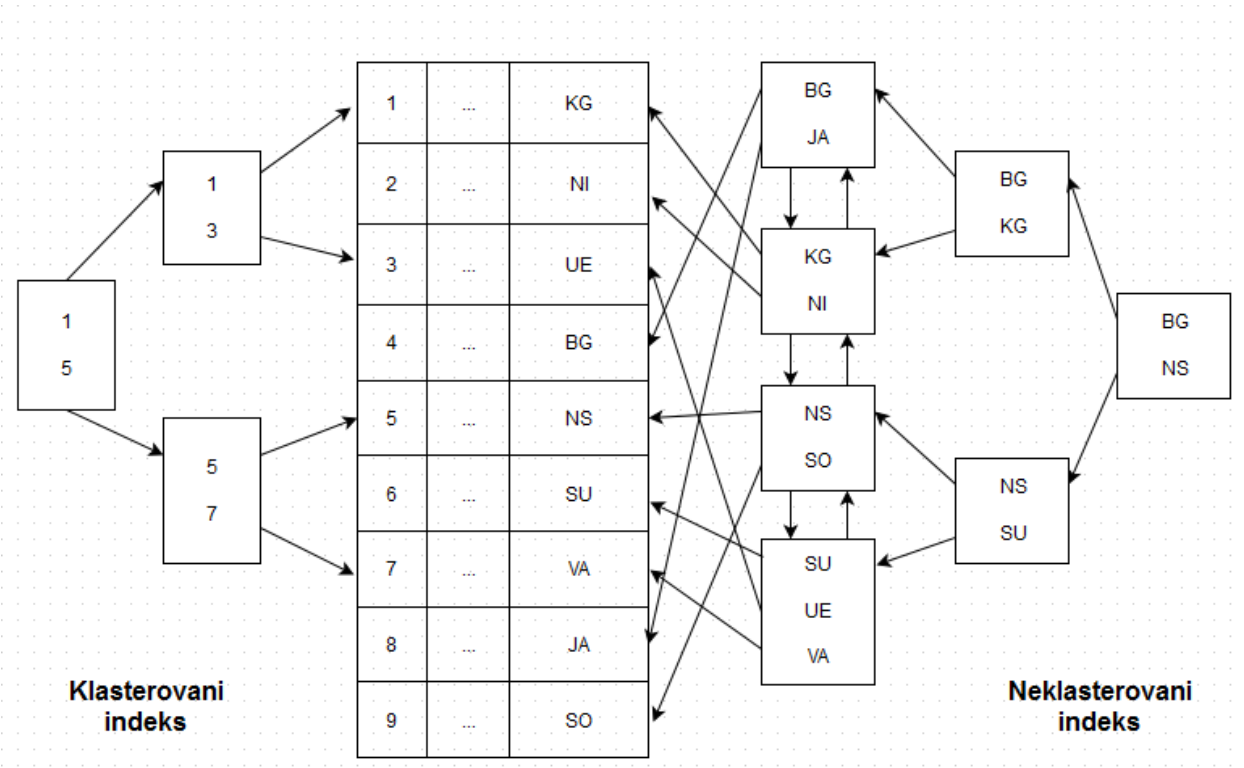
Slika 1 – Primer B-stabla



Slika 2 – Struktura indeksa

Kreiranje klasterovanog indeksa podrazumeva fizičko sortiranje tabele po ključu pretrage, a zatim izgradnju B-stabla nad stranicama tabele. Ovde stranice tabele mogu da se posmatraju kao listovi. Pretragom kroz stablo se dolazi u blizinu traženog podatka, tačnije učitava se stranica koja ga sadrži. Zatim se prolazi red po red kroz stranicu dok se ne nađe tražena vrsta. Tabela sa klasterovanim indeksom više nije uređena kao hip, već sekvencijalno, stranice su poređane po vrednosti ključa pretrage. Ključ klasterovanog indeksa je najčešće primarni ključ, ali ne mora nužno biti tako. Nad jednom tabelom može biti napravljen samo jedan klasterovani indeks jer tabela može biti sortirana samo po jednom ključu.

Neklasterovani indeksi u listovima imaju sve vrednosti ključa pretrage koje se pojavljuju u tabeli. Te vrednosti su u indeksu sortirane po ključu pretrage i svaka vrednost referencira odgovarajući zapis u tabeli. Neklasterovani indeksi ne utiču na fizički raspored podataka. Zbog toga ih može biti više nad jednom tabelom.



Slika 3 – Struktura klasterovanog i neklasterovanog indeksa

Klasterovani indeks se može uporediti sa telefonskim imenikom, a neklasterovani sa indeksom pojmova na kraju knjige. U telefonskom imeniku su zapisi sortirani i pretragom po slovima se brzo može doći do tražene osobe. Indeks pojmova na kraju knjige daje sortiran spisak pojmova sa informacijom o tome na kojim stranicama se pojavljuje taj pojam. Na sličan način funkcionišu i klasterovani i neklasterovani indeks.

Query optimizer razmatra upotrebu indeksa pri izvršavanju upita sa WHERE, GROUP BY, ORDER BY i JOIN naredbama. Indeksi se najčešće prave nad kolonama koje se pojavljuju kao argumenti pretrage u WHERE naredbi, ili kao spoljašnji ključ u JOIN naredbi. Klasterovani indeksi se obično postavljaju nad nepromenljivim kolonama, a neklasterovani tamo gde se bira mali procenat podataka iz tabele.