Interna struktura i organizacija indeksa MSSQL baze podataka

Seminarski rad

Strahinja Laktović
br. indeksa 1089

Sistemi za upravljanje bazama podataka
Elektronski fakultet u Nišu

1.Uvod. 3
2.Indeksi i strukture podataka indeksa....4
3.Klasifikacija indeksa....5
4.Tipovi indeksa MS SQL Server-a....5
4.1. Heš indeksi....5
4.2. Klasterizovani indeksi....6
4.3. Neklasterizovani indeksi....8
4.4. Jedinstveni indeksi....8
4.5. Indeksi filtriranja....9
4.6. Indeksi sa dodatnim kolonama koje nisu ključevi....9
4.7. Indeksi nad kolonama koje se računaju....9
4.8. Columnstore indeksi....9
4.9. Prostorni indeksi....10
4.10. XML indeksi....10
4.11. Full-text indeksi....11
5.Primeri korišćenja indeksa i njihovih performansi u optimizaciji upita....11
5.1. Uporedjiivanje upita s klasterizovanim indeksom i bez indeksa....12
5.2. Testiranje upita sa neklasterizovanim indeksom....13
5.3. Testiranje upita sa neklasterizovanim indeksom i filtriranim neklasterizovanim indeksom....14
5.4. Testiranje upita sa neklasterizovanim indeksom i columnstore indeksom....15
6. Zaključak....17
Literatura 18

1. Uvod

Baze podataka, zajedno sa sistemima za njihovo upravljanje, koriste se za skladištenje, čitanje i obradu podataka koji se čuvaju putem različitih aplikacija. Zbog fizičke prirode medijuma za skladištenje podatka i njihovih ograničenja oduvek je bio imperativ nalaženje načina za efikasno i brzo čitanje baza i obradu podataka.

U cilju rešavanja ovih problema koriste se posebne tehnike za skladištenje i pretraživanje. Indeksiranje je jedna od glavnih tehnika koja, u nekim slučajevima, organizuje skladištenje podataka i povećava brzinu čitanja i obrade podataka.

Ovaj rad će se baviti tehnikama indeksiranja i klasifikacijom indeksa. Internom organizacijom, implementacijom i najboljim praksama za korišćenje indeksa kod MS SQL sistema za upravljanje bazama podataka. U zadnjem delu će biti prikazani primeri korišćenja indeksa i njihov uticaj na performanse upita kod MS SQL-a.

U drugom poglavlju biće reči o organizaciji podataka na disku, motivaciji za korišćenje indeksiranja, indeksima i strukturama podataka koje se koriste pri izradi indeksa.

Treći deo se bavi klasifikacijom indeksa po raznim kriterijumima.

Vrste indeksa u MS SQL-u, njihova implementacija, interna struktura i preporuke pri dizajniranju indeksa za rešavanje različitih problema koje specifičnost šeme baze podataka stvara jesu tema četvrtog poglavlja.

Peti deo poredi brzine upita, korišćenjem stvarne baze podataka, sa i bez indeksa u primerima koji opravdavaju korišćenje specifične vrste indeksa.

2. Indeksi i strukture podataka indeksa

Diskovi su najvažniji uredjaji za skladištenje kod sistema za upravljanje bezama podataka. Kolekcija torki koje se čuvaju na fajlu jesu stranice, a fajlovi se sastaje od jedne ili više stranica i čine osnovnu apstrakciju sloja skladištenja podataka. Svaka torka ima identifikator pomoću kog je moguće naći adresu na disku na kojoj se nalazi. Torke su najčešće nasumično razbacane po različitim stranicama fajla. Zbog toga je moguće pribaviti sve torke iz fajla ili odredjene torke pomoću identifikatora.

Indeksi su fajlovi sa specifičnom strukturom podataka koja povećava brzinu čitanja grupe torki podataka po ceni dodatnih upisa i održavanje strukture indeksa. Bez indeksa je potrebno pristupiti svakoj stranici fajla da bi se proverilo da li je tamo sačuvana torka koja zadovoljava dati uslov, što je vrlo neoptimalno jer su U/I operacije nekoliko reda veličine skuplje operacije od ostalih pri čitanju podataka. Indeksiranje torki se vrši po odredjenom ključu pretrage. Korišćenjem ključa pretražuje se indeks struktura.

Indeksi se mogu koristiti na jedan od tri načina:

- Kao poseban način organizacije fajla, čime se postiže sortiranje torki u fajlu po ključu pretrage indeksa. U ovom slučaju se u indeks fajlu nalaze torke.
- Kao fajl sa ključ/vrednost parovima gde je ključ ključ pretrage indeksa a vrednost identifikator torke
- Na sličan način kao i prethodni, s tim što je sada vrednost u ključ/vrednost parovima zapravo niz identifikatora torki

Drugi i treći način korišćenja su nezavisni od organizacije fajla koji se indeksira (onog koji sadrži torke). Da bi se izbeglo dupliranje podataka, ukoliko se vrši indeksiranje torki po više ključeva onda se prvi način indeksiranja može koristiti samo za jedan ključ pretrage.

Dva osnovne strukture podataka koje se koriste pri indeksiranju jesu heš tablice i stabla pretrage.

Indeksiranje bazirano na heširanju koristi heš funkciju ključa da pronadje podatke koji imaju odredjenu vrednost ključa. Podaci u fajlu su organizovani u grupe (eng. *bucket*) stranica. Jedna grupa sadrži jednu glavnu stranicu i više drugih, lančano povezanih stranica, ukoliko su potrebne.

Da bi se pristupilo fajlovima koji imaju vrednost ključa K potrebno je primeniti heš funkciju nad ključem. Rezultat funkcije je identifikator grupe gde treba pretražiti sve stranice sa torkama grupe.

Pri dodavanju novog podatka vrši se heš funkcija nad atributom po kom se indeksira na osnovu koje se on dodaje u odgovarajuću grupu. Prilikom dodavanja se alociraju dodatne stranice ukoliko su potrebne.

Sa druge strane, za pretragu podataka po nekom drugom atributu koji nije ključ indeksa potrebno je skeniranje čitavog fajla.

Strukture bazirane na stablu pretrage sadrže podatke sortirane po vrednosti ključa i održavaju hijerarhijsku strukturu podataka za pretragu po ključu. Stablo je organizovano tako da su listovi stabla stranice sa podacima a ostali čvorovi sadrže pointere na podstablo odvojene vrednošću ključa K. Pointer levo od K ukazuje na podstablo gde se nalaze podaci kojima je vrednost ključa manja od K. Pointer desno od K ukazuje na podstablo podataka čija je vrednost

ključa veća ili jednaka K. Ukoliko se u pretrazi podataka specificira raspon vrednosti koje ključ može da ima česta je pojava da se podaci nalaze na više stranica. Zbog toga su stranice, listovi, organizovani u dvostruko spregnute lančane liste. Broj U/I operacija potrebnih za pristupanje podacima jednak je zbiru visine stabla i broju stranica na koje se sve oni nalaze. Stablo pretrage je znatno brže od binary search pretrage zato što svaki čvor sadrži više vrednosti i visina stabla je praktično ne veća od četiri.

Praktičnu primenu imaju i strukture bazirane na heš funkcijama i one bazirane na stablima pretrage. Heš indeksi se koriste kada se unapred zna da će se ključevi koristi u relacijama jednakosti pri pretrazi. Sa druge strane ukoliko se traže podaci čije vrednosti ključa pretrage pripadaju nekom rasponu onda je moguće koristiti samo stabla pretrage.

3. Klasifikacija indeksa

Kada je fajl organizovan tako da je redosled podataka u njemu isti ili približno isti redosledu podataka u indeksnoj strukturi onda je indeks *klasterizovan*. To je uvek slučaj sa vrstom indeksa gde se indeks za organizaciju samog fajla sa podacima. Kod pristupa gde su indeksi posebni fajlovi sa parovima ključ/idenktifikator(i) podataka indeksu su uglavnom *neklasterizovani*, osim ako fajlovi nisu sortirani po ključu pretrage, što najčešće nije slučaj.

Indeksi koji za ključ pretrage imaju polja koja uključuju primarni ključ jesu *primarni* indeksi. Svi ostali indeksi su *sekundarni*.

Indeksi u strukturi mogu da sadrže više čvora koji upućuju na isti podatak. Primarni indeksi i sekundarni indeksi čiji je ključ pretrage atribut koji je jedan od ključeva kandidata nemaju duplikate i oni se zovu *jedinstvenim*.

Ključevi indeksa se mogu sastojati i od kompozicije vrednosti većeg broja kolona. Indeksi sa ovakvim ključevima zovu se *kompozitni* indeksi.

4. Tipovi indeksa MS SQL Server-a

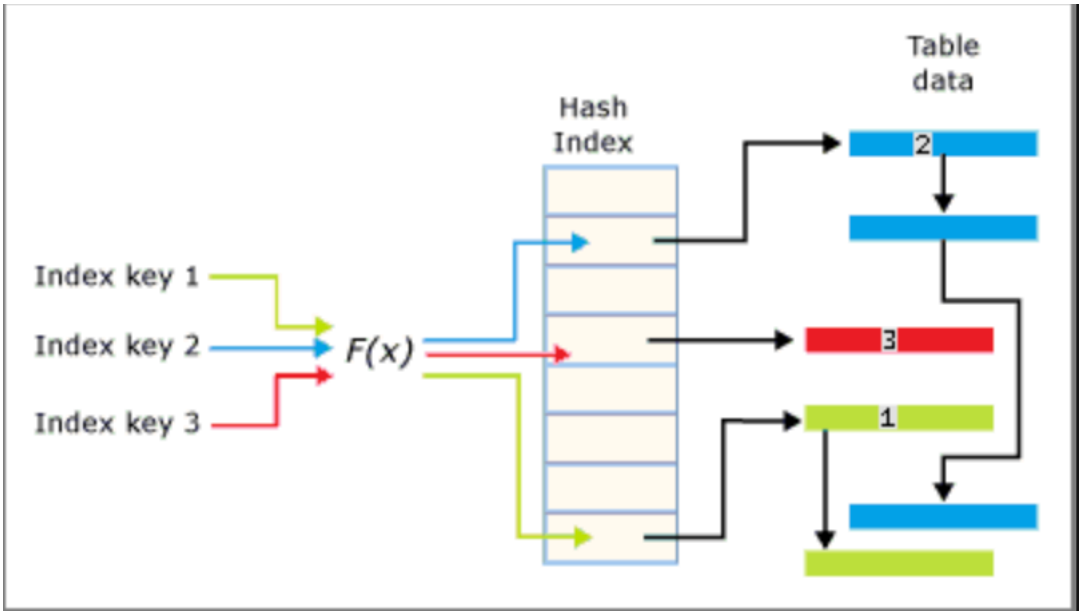
4.1 Heš indeksi

Heš indeksi se u SQL Serveru koriste samo kod memorijski optimizovanih OLTP baza podataka. Kod ovih baza podataka podaci su smešteni u memoriji i ne čuvaju se na disku. Heš indeksi koriste koriste fiksnu količinu memorije koja se može izračunati kao funkcija broja grupa (eng. *bucket*).

Heš indeks čini niz pointera koji ukazuju na grupe podataka. Grupe sadrže parove ključ/ vrednost gde je ključ takav da se njegovim heširanjem dobije pointer na grupu a vrednost je adresa reda memorijsko-optimizovanoj tabeli.

Heš funkcija u SQL Server-u je uvek ista za sve heš indekse, deterministička i balansirana. Broj grupa heš indeksa se definiše pri definisanju indeksa. Potrebno je pažljivo izabrati broj grupa zato što mali broj grupa može dovesti do većeg broja kolizija, što ima posledicu za to da lanci jedne grupe budu veliki i da zbog toga pretraga može biti sporija. Sa druge strane veliki

broj grupa može dovesti do toga da neke budu prazne i zauzimaju memoriju, što usporava full scan pretrage.



4.1 Korišćenje heš funkcije za preslikavanje ključeva na pointere grupa redova **[4]**

Heš indeksi su dobri u situacijama kad se u SQL upitima u WHERE klauzuli koriste tačne vrednosti za kolone za koje je definisan indeks.

Imaju loše performanse u situacijama gde se specificira raspon vrednosti za kolone za koje je definisan indeks ili su specificirane tačne vrednosti samo za neke od kolona, a ne za sve, kad je u pitanju kompozitni indeks.

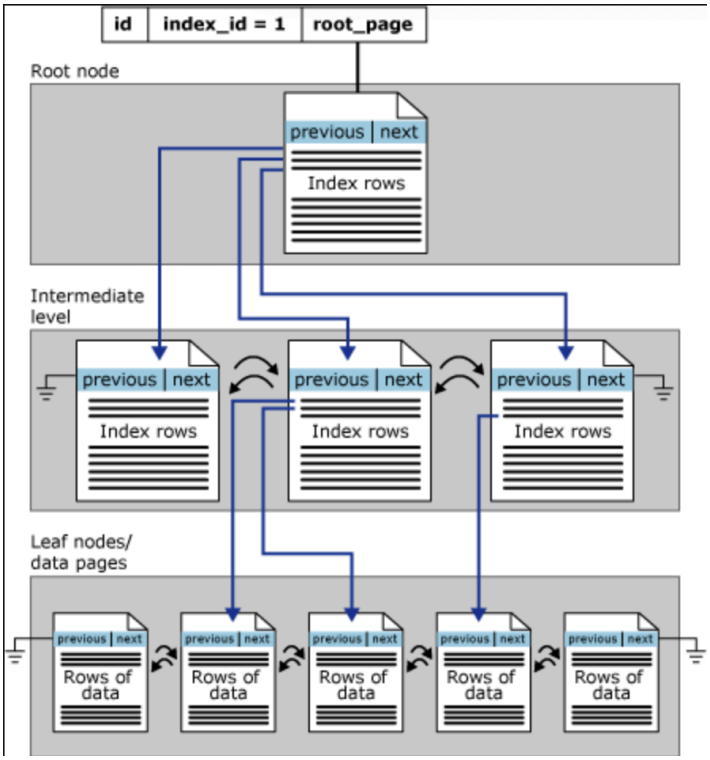
4.2 Klasterizovani indeksi

Klasterizovani indeksi sortiraju redove u tabeli na osnovu vrednosti ključa. Moguće je kreirati samo jedan klasterizovan indeks po tabeli zato što se podaci mogu sortirati samo po jednom parametru. Klasterizovan indeks se najčešće pravi nad primarnim ključem, običnim ili kompozitnim, ili nad kolonom ili kolonama koje jedinstveno odredjuju red u tabeli. Koriste se za u upitima koji se ponavljaju često, a pogodni su i za upite koji imaju raspon vrednosti za ključ, zbog sortiranja.

Kada se definiše primarni ključ nad tabelom database engine uvek u pozadini kreira klasterizovan, jedinstven indeks nad njim.

Klasterizovani indeksi su u SQL Serveru organizovani kao B stabla. Koren i čvorovi na u sredini stabla imaju vrednosti ključa i pointere na podstabla levo i desno od ključa koji sadrže vrednosti ključa manje odnosno veće ili jednake vrednosti od ključa, respektivno. Listovi sadrže pointere na same podatke u tabeli.

Svi klasterizovani indeksi imaju jedan red u *sys.partitions* tabeli za svaku particiju koju imaju. *sys.partitions* tabela sadrži polje *index_id* koje ima vrednost 1 za klasterizovane indekse i polje *hobt_id* sa pointerom na koren stabla indeksa. Takodje, u zavisnosti od tipa podataka koji se čuvaju u indeksu, indeks alocira dodatne alokacione jedinice u bazi za čuvanje podataka strukture.



4.2 Struktura B stabla kod klasterizovanih indeksa **[4]**

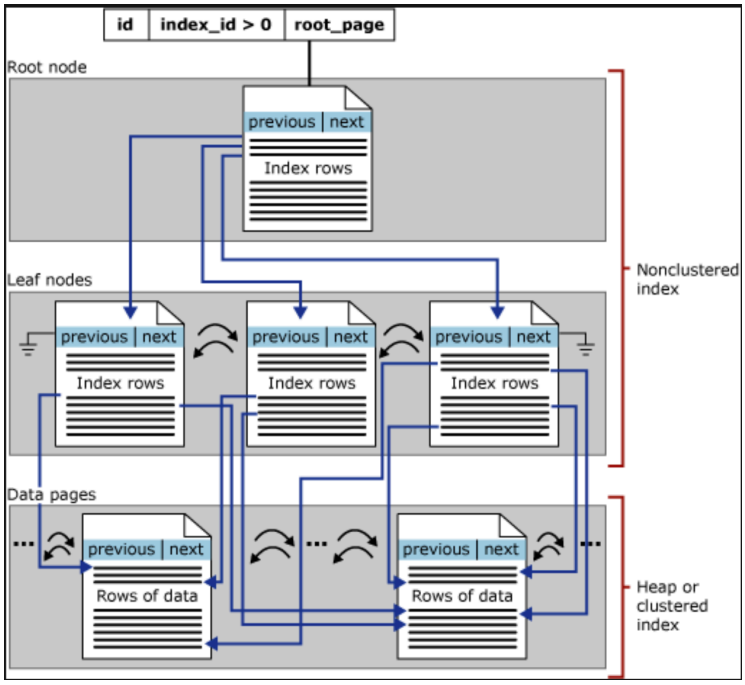
Klasterizovane indekse treba koristiti u slučaju traženja rezultata po ključu sa operatorima uporedjivanja i BETWEEN. Upiti sa JOIN-om takodje bivaju brži zato što se JOIN vrši na osnovu stranog ključa. Kod upita sa GROUP BY i ORDER BY klauzulama, ukoliko se one vrše nad ključevima klasterizovanog indeksa onda database engine neće dodatno sortirati podatke po ključu.

Klasterizovani indeksi su dobri za kolone sa jedinstvenim, ili bar približno jedinstvenim vrednostima, ukoliko se podacima pristupa sekvencijalno po ključu indeksa ili se stalno sortiraju po koloni po kojom se indeksira.

Vrednosti ključeva kod klasterizovanih indeksa se koriste u listovima svih neklasterizovanih indeksa. Zato nije dobro korišćenje velikih kompozitnih ključeva Kolona čije se vrednosti koriste za indeks ne treba da biti ona čije se vrednosti stalno menjaju jer se onda vrši ponovno sortiranje fajla i održavanje indeks strukture pri svakoj promeni.

4.3 Neklasterizovani indeksi

Neklasterizovani indeksi se mogu napraviti nad tabelom ili pogledom koja ima klasterizovan indeks ili je neorganizovana gomila podataka (eng. *heap*). Struktura neklasterizovanih indeksa jeste B stablo. Ono se razlikuje od stabla klasterizovanog indeksa po tome što listove čine indeks stranice. One imaju redove sa ključ/vrednost parovima gde je ključ vrednost neklasterizovanog indeksa a vrednost ključ klasterizovanog indeksa za taj red u tabeli ili jedinstveni identifikator reda u hrpi.



4.3 Struktura B stabla kod neklasterizovanih indeksa **[4]**

Neklasterizovane indekse je dobro napraviti nad kolonama koje se koriste pri korišćenju klauzula JOIN i GROUP BY, pritom klasterovan indeks treba da postoji nad stranim ključevima korišćenim u JOIN-u. Ukoliko je broj podataka koji se vraćaju mali klasterizovani indeks može biti veoma pogodan. Kada je broj podataka veliki tada neklasterizovan indeks nema dobre performanse zato što podaci nisu sortirani po ključu neklasterizovanog indeksa i može dovesti do velikog broja U/I operacija kao kod skeniranja cele tabele.

Dalje optimizacije nad upitima za koje je dobro koristiti neklasterizovane indekse mogu se izvršiti korišćenjem indeksa filtriranja ili neklasterizovanih indeksa sa dodatnim kolonama koje nisu ključevi, o kojima će biti reči kasnije.

4.4 Jedinstveni indeksi

Jedinstveni indeksi su takvi da nemaju duplikate, što znači da je svaki red u tabeli ili pogledu koji je indeksiran na neki način jedinsten. Mogu biti i klasterizovani i neklasterizovani.

4.5 Indeksi filtriranja

Indeksi filtriranja su optimizovani neklasterizovani indeksi gde se pre indeksiranja vrši filtriranje ključeva kolona po nekom predikatu. Ovim se postiže ušteda pri održavanju i skladištenju indeksa. Filtrirana statistika koja se koristi pri optimizaciji kverija je preciznija od statistike cele tabele.

Indekse filtriranja je dobro upotrebiti kada kolone koje se indeksiraju imaju mali broj NULL vrednosti, sadrže kategoričke podatke ili vrednosti u odredjenom rasponu kao što su datumi i vreme. Ukoliko je predikat filtriranja kompleksan i podrazumeva korišćenje više tabela onda treba napraviti view umesto indeksa filtriranja.

4.6 Indeksi sa dodatnim kolonama koje nisu ključevi

Neklasterizovani indeksi mogu imati dodatne kolone koje nisu ključevi na nivou listova i time se iskoristiti mogu iskoristiti kolone sa tipovima podataka koji nisu dozvoljeni kao ključevi indeksa. Database engine ne računa ove kolone kada računa veličinu indeks ključa i potrebnu memoriju za održavanje indeksa ali zato čita ove vrednosti i znatno utiče na brzinu izvršenja upita kada se u WHERE klauzuli sve navedene kolone nad kojima se specificira neka vrsta filtera nalaze u listu indeksa kao ključ ili kao dodatna kolona.

Neklasterizovani kompozitni ključevi se trebaju pretvoriti u indekse sa dodatnim kolonama gde kao ključevi ostaju kolone samo po kojima se pretražuju podaci. Generalno neklasterizovani kompozitni ključevi mogu da imaju velike strukture podataka koje je teško održavati. Ovom transformacijom se ta struktura i njena cena održavanja smanjuju. Kolone s tipovima podataka *text, ntext* i *image* ne mogu biti dodatne kolone.

4.7 Indeksi nad kolonama koje se računaju

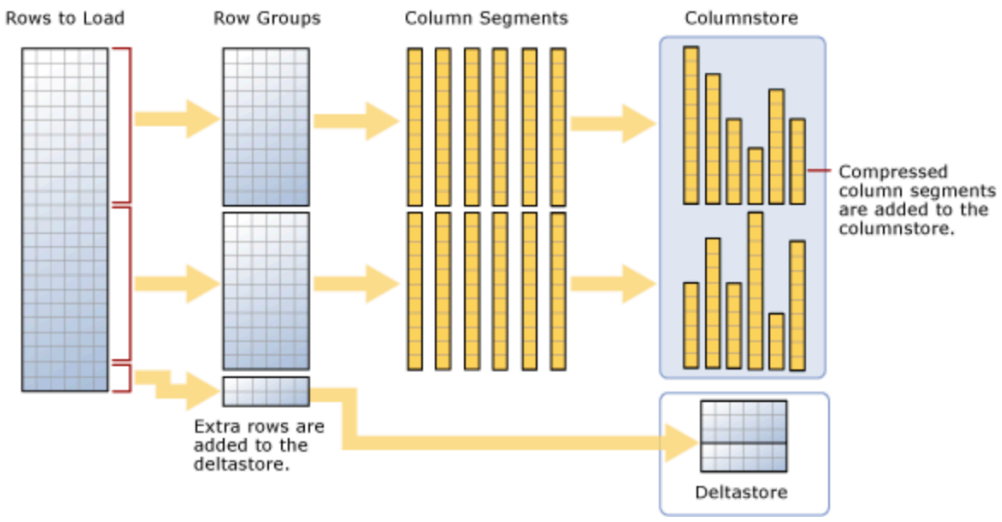
Ukoliko se vrednosti u nekoj koloni računaju kombinacijom vrednosti drugih kolona ili uz pomoć determinističkog unosa onda je ta kolona takodje pogodna za indeksiranje. Ovakve kolone se mogu koristiti i kao dodatne kolone neklasterizovanih indeksa dok god rezultat računanja nije tipa *text*, *ntext* i *image.*

4.8 Columnstore indeksi

Columnstore indeksi su posebna vrsta indeksa, pogodna za podatke koje se čuvaju u skladištima podataka koji primarno imaju *grupne insert* operacije i *read-only* upite.

Kod columnstore-a indeksa podaci se čuvaju drugačije u odnosu na podatke kojima se pristupa pomoću indeksa koji imaju strukturu B stabla koji se drugačije mogu nazvati i *rowstore* indeksi. Podaci koji imaju minimum 102,400 reda i maksimalno 1 million, se grupišu u *grupe redova*. Svaka grupa redova se segmentiše po kolonama. Ti *segmenti kolona* su osnovna jedinica čuvanja columnstore indeksa. Oni redovi koji ostanu posle formiranja grupa redova i ima ih premalo da formiraju svoju grupu čuvaju se kao *deltastore,* struktura koja je organizovana u

obliku B stabla kao klasični indeksi. Postoji poseban proces - *tuple mover* koji radi u pozadini i vrši prebacivanje redova iz deltastore u rowstore ukoliko bude bilo potrebe.



***4.4 Transformacija podataka koji se učitavaju u bazu u columnstore* [5]**

Ono što čini columnstore indekse efikasnim je to da se podaci smešteni po segmentima kolona efikasno skladište i kompresiju. *Batch mode* obrada podataka SQL servera omogućava optimizovanu, paralelnu obradu podataka koji su sačuvani u obliku columnstore indeksa.

Columnstore indeksi mogu biti biti klasterizovani i neklasterizovani. Klasterizovani columnstore indeksi služe kao primarna organizacija podataka i koriste se kod skladišta podataka. Neklasterizovani columnstore indeksi funkcionišu isto kao klasterizovani ali su sekundarni indeksi nad rowstore podacima. Oni su pogodni za real-time analitiku kod kod OLTP sistema.

4.9 Prostorni indeksi

Prostorni indeksi su indeksi organizovani kao B stabla i omogućavaju indeksiranje kolona tipa *geometry*.

4.10 XML Indeksi

XML indeksi su indeksi koji se kreiraju nad kolonama tipa *XML*. Koriste se kada se u aplikaciji često koriste upiti nad XML kolonama, ili kad su XML vrednosti velike a potrebni su njihovi delovi koji su relativno mali. Postoje primarni i sekundarni XML indeksi.

4.11 Full-text indeksi

Indeksi bazirani na tokenima koji omogućava naprednu pretragu reči u podacima koji sadrže velike tekstove.

5. Primeri korišćenja indeksa i njihovih performansi u optimizaciji upita

Za potrebe testiranja indeksa kreiran je SQL Server 2019 pomoću docker slike *mcr.microsoft.com/mssql/server:2019-latest.*

Uz pomoć ASP.NET Core aplikacije izgenerisano je 500,000 redova u tabeli *Users* i 2,200,000 redova u tabeli *WarehouseStatistics*. Šeme ovih tabela prikazane su na slikama 5.1 i 5.2.

```
CREATE TABLE [dbo].[WarehouseStatistics](
    [Id] [int] IDENTITY(1,1) NOT NULL,
    [ItemName] [nvarchar](200) NULL,
    [Count] [int] NOT NULL
) ON [PRIMARY]
```

5.1 Šema tabele WarehouseStatistics

```
CREATE TABLE [dbo].[Users](
    [Id] [int] IDENTITY(1,1) NOT NULL,
    [Name] [nvarchar](200) NULL,
    [LastName] [nvarchar](200) NULL,
    [TrainingCompletionDate] [datetime2](7) NULL
) ON [PRIMARY]
```

5.2 Šema tabele Users

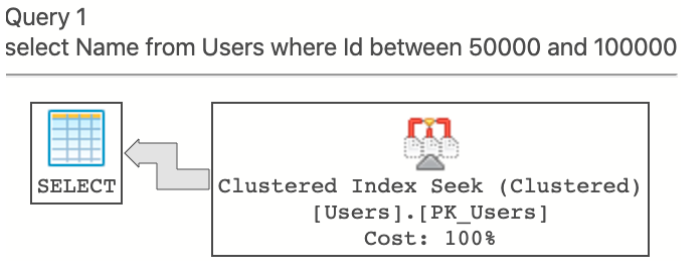
Sve upite koji će se izvršiti nad bazom u cilju demonstracije upotrebe indeksa pratiće dve slike.

Prva slika je slika na kojoj će se videti sam upit i plan izvršenja upita koji pokazuje da li se koriste indeksi u procesu pribavljanja podataka, i koji.

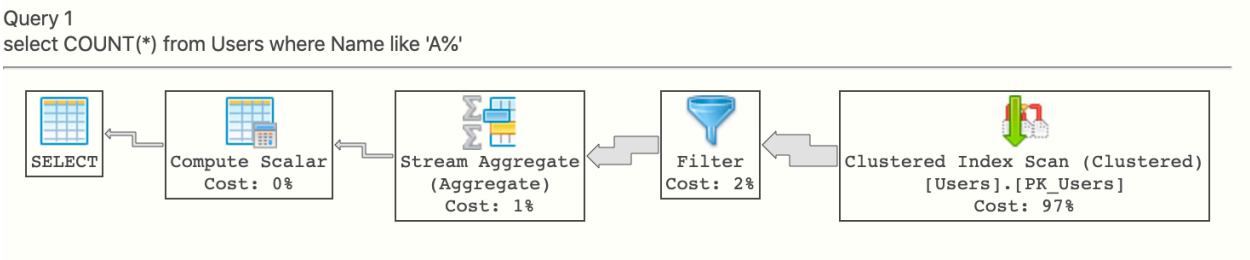
Druga slika je slika informacija o upitu gde će zelenom bojom biti podvučeno vreme zadnje I/O operacije koje će varirati od vrste indeksa koji se koristi ili ne u upitu. Crvenom bojom će biti podvučen broj redova koji je pročitan iz tabele. Ovaj broj je značajan zato što će se koristiti upiti koji pravilnom upotrebom indeksa čitaju manji broj redova, a nepravilnom upotrebom čitaju čak i sve redove.

5.1 Uporedjiivanje upita s klasterizovanim indeksom i bez indeksa

Prvi upit pribavlja imena ljudi kojima je Id izmedju 50,000 i 100,000. Drugi upit nalazi broj ljudi čije ime počinje na slovo A. Iako oba upita rade nad otprlike 50,000 redova u tabeli na slikama 5.3, 5.4 i 5.5 videćemo da je za potrebe prvog upita korišćen klasterizovan indeks nad id-jem i broj pročitanih redova 50,001, dok je u drugom upitu moralo biti izvršeno skeniranje cele tabele jer nema indeksa nad kolonom Name. Broj pročitanih redova je svih 500,000 a dužina trajanja U/I operacije deset puta veća.



5.3 Plan izvršenja upita koji koristi klasterizovani indeks



5.4 Plan izvršenja upita koji ne koristi klasterizovani indeks

Clustered Index Seek (Clustered)
Scanning a particular range of rows from a clustered index.

Physical Operation	Clustered Index Seek
Logical Operation	Clustered Index Seek
Estimated Execution Mode	Row
Storage	RowStore
Estimated Operator Cost	0.361246 (100%)
Estimated I/O Cost	0.306088
Estimated CPU Cost	0.0551581
Estimated Subtree Cost	0.361246
Estimated Number of Executions	1
Estimated Number of Rows to be Read	50001
Estimated Number of Rows	50001
Estimated Row Size	4035 B
Ordered	True
Node ID	0

Output List
[IndexesTest].[dbo].[Users].Name
Object

Clustered Index Scan (Clustered)
Scanning a clustered index, entirely or only a range.

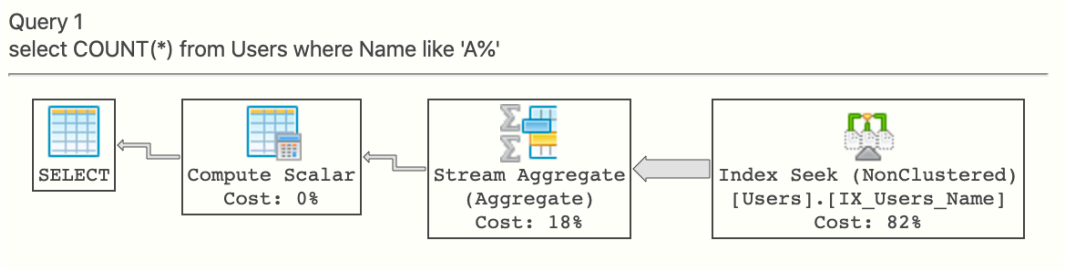
Physical Operation	Clustered Index Scan
Logical Operation	Clustered Index Scan
Estimated Execution Mode	Row
Storage	RowStore
Estimated Operator Cost	3.58662 (97%)
Estimated I/O Cost	3.03646
Estimated CPU Cost	0.550157
Estimated Subtree Cost	3.58662
Estimated Number of Executions	1
Estimated Number of Rows to be Read	500000
Estimated Number of Rows	500000
Estimated Row Size	4035 B
Ordered	False
Node ID	3

Output List
[IndexesTest].[dbo].[Users].Name
Object

5.5 Levo na slici je cena U/I operacije i broj pročitanih redova upita koji koristi klasterizovani indeks, a desno upita koji ne koristi indeks

5.2 Testiranje upita sa neklasterizovanim indeksom

Upit na slikama 5.6 i 5.7 je isti kao i upit u prethodnom testu gde se nalazi broj ljudi kojima ime počinje na slovo A. Sada, medjutim, postoji neklasterizovan indeks nad kolonom Name. Vidimo da je trajanje U/I operacije *2.5 puta manje* sada kada postoji indeks, Broj pročitanih redova je približno 50,000, koliki je i broj ljudi.



5.6 Plan izvršenja upita koji koristi neklasterizovani indeks

Index Seek (NonClustered)
Scan a particular range of rows from a nonclustered index.

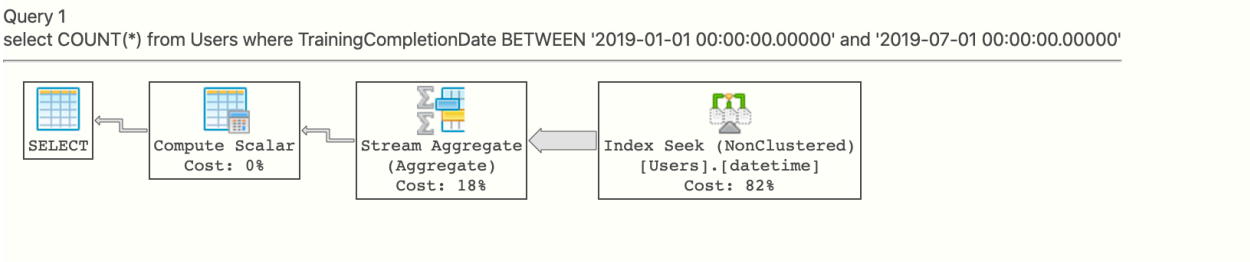
Physical Operation	Index Seek
Logical Operation	Index Seek
Estimated Execution Mode	Row
Storage	RowStore
Estimated Operator Cost	0.176564 (82%)
Estimated I/O Cost	0.121644
Estimated CPU Cost	0.0549208
Estimated Subtree Cost	0.176564
Estimated Number of Executions	1
Estimated Number of Rows to be Read	49785.3
Estimated Number of Rows	49785.3
Estimated Row Size	211 B
Ordered	True
Node ID	2

Object
[IndexesTest].[dbo].[Users].[IX_Users_Name]
Seek Predicates

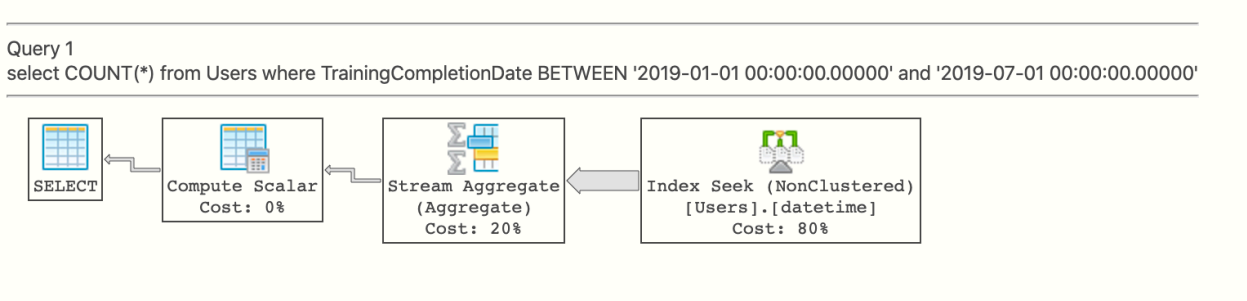
5.7 Cena U/I operacije i broj pročitanih redova upita koji koristi neklasterizovani indeks

5.3 Testiranje upita sa neklasterizovanim indeksom i filtriranim neklasterizovanim indeksom

Testiranje je obavljeno korišćenjem upita koji vraća broj ljudi čiji se trening završio u prvoj polovini 2019. godine. Prvi upit je odradjen nakon postavljanja neklasterizovanog indeksa nad kolonom CompletionTrainingDate. Isti taj upit je ponovljen nakon stavljanja neklasterizovanog indeksa samo nad podacima kod kojih CompletionTrainingDate nije NULL. Tačno 80% podataka ima definisan ovaj datum. Uporedjivanjem ova dva upita vidimo da je trajanje U/I operacije izvršenja upita sa indeksom bez filtera *25% duže* od upita sa filtriranim indeksom.



5.8 Plan izvršenja upita koji koristi nefiltrirani neklasterizovani indeks



5.9 Plan izvršenja upita koji koristi filtrirani neklasterizovani indeks

Index Seek (NonClustered)
Scan a particular range of rows from a nonclustered index.

Physical Operation	Index Seek
Logical Operation	Index Seek
Estimated Execution Mode	Row
Storage	RowStore
Estimated Operator Cost	0.552496 (82%)
Estimated I/O Cost	0.332755
Estimated CPU Cost	0.219741
Estimated Subtree Cost	0.552496
Estimated Number of Executions	1
Estimated Number of Rows to be Read	199622
Estimated Number of Rows	199622
Estimated Row Size	9 B
Ordered	True
Node ID	2

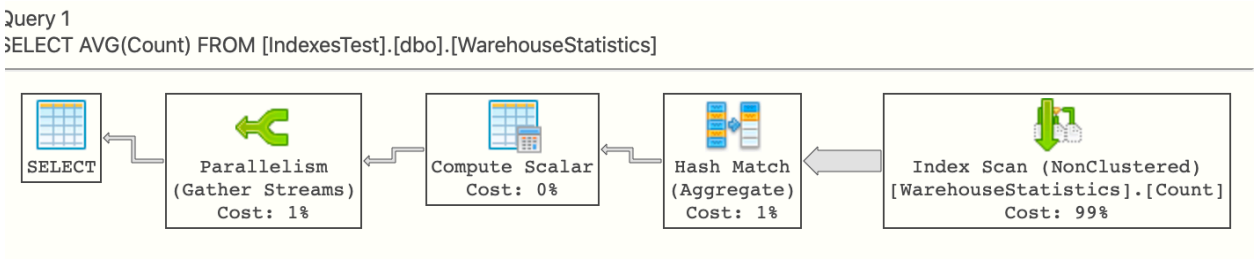
Index Seek (NonClustered)
Scan a particular range of rows from a nonclustered index.

Physical Operation	Index Seek
Logical Operation	Index Seek
Estimated Execution Mode	Row
Storage	RowStore
Estimated Operator Cost	0.486558 (80%)
Estimated I/O Cost	0.266829
Estimated CPU Cost	0.219729
Estimated Subtree Cost	0.486558
Estimated Number of Executions	1
Estimated Number of Rows to be Read	199611
Estimated Number of Rows	199611
Estimated Row Size	9 B
Ordered	True
Node ID	2

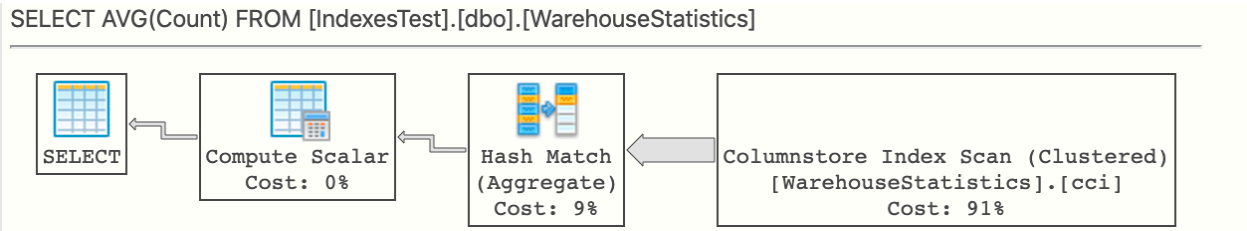
5.10 Levo na slici je cena U/I operacije i broj pročitanih redova upita koji koristi nefiltrirani neklasterizovani indeks, a desno upita koji koristi filtrirani indeks

5.4 Testiranje upita sa neklasterizovanim indeksom i columnstore indeksom

Upit je radjen nad tabelom WarehouseStatistics koja ima preko 2 miliona redova da bi se napravila bar dva segmenta kolona pune veličine, i jedan manji kada se koristi columnstore indeks. Upit je tražio prosečnu vrednost polja Count. Korišćenjem neklasterizovanog indeksa umesto columnstore-a videćemo da je cena zadnje U/I operacije približno *95 puta* veća. Oviim se uočava ogromna prednost columnstore indeksa u obradi ovog tipa podataka. Rezultati izvršenja prikazani su na slikama 5.11, 5.12 i 5.13



5.11 Plan izvršenja upita koji koristi neklasterizovani indeks



5.12 Plan izvršenja upita koji koristi columnstore indeks

Index Scan (NonClustered)

Physical Operation	Index Scan
Logical Operation	Index Scan
Estimated Execution Mode	Batch
Storage	RowStore
Estimated Operator Cost	3.63133 (99%)
Estimated I/O Cost	2.82461
Estimated CPU Cost	0.806719
Estimated Subtree Cost	3.63133
Estimated Number of Executions	1
Estimated Number of Rows to be Read	2.2e+06
Estimated Number of Rows	2.2e+06
Estimated Row Size	11 B
Ordered	False
Node ID	3

Output List
[IndexesTest].[dbo].[WarehouseStatistics].Count
Object
[IndexesTest].[dbo].[WarehouseStatistics].[Count]

Columnstore Index Scan (Clustered)
Scanning a clustered index, entirely or only a range.

Physical Operation	Clustered Index Scan
Logical Operation	Clustered Index Scan
Estimated Execution Mode	Batch
Storage	ColumnStore
Estimated Operator Cost	0.245141 (91%)
Estimated I/O Cost	0.003125
Estimated CPU Cost	0.242016
Estimated Subtree Cost	0.245141
Estimated Number of Executions	1
Estimated Number of Rows to be Read	2.2e+06
Estimated Number of Rows	2.2e+06
Estimated Row Size	11 B
Ordered	False
Node ID	2

Output List
[IndexesTest].[dbo].[WarehouseStatistics].Count

5.13 Levo na slici je cena U/I operacije i broj pročitanih redova upita koji koristi neklasterizovani indeks, a desno upita koji koristi columnstore indeks

6. Zaključak

Indeksiranje je jedna od najmoćnijih tehnika ubrzavanja pribavljanja i obrade podataka iz baze. Postoji mnogo različitih tehnika indeksiranja.

Najveće grupe indeksa baziraju se na heširanju i stablima pretraga, tehnikama koje višestruko ubrzavaju pretragu po ključevima.

U ovom radu izvršena je klasifikacija indeksa na osnovu sortiranosti podataka u odnosu na ključ pretrage, toga da li je ključ primarni ključ tabele ili neka druga kolona, i po jedinstvenosti ključeva u koloni koja se indeksira kao i mogućnosti da se ključevi indeksa prave od više od jedne kolone u tabeli.

Veći deo indeksa u MS SQL Server sistemu za upravljanje bazama podataka baziran je na B stablima. Oni koji nisu se odnose na specijalne tipove kolona kao što su kolone sa geometrijskim podacima i XML podacima.

Posebno zanimljiv tip indeksa jeste columnstore indeks koji je potpuno drugačiji od ostalih i koji obradu posebne vrste podataka vrši nekoliko reda veličine brze od klasičnih indeksa.

U zadnjem poglavlju rada demonstrirane su razne tehnike indeksiranja, tehnike čijim se uporedjivanjem može zaključiti da razne vrste indeksa imaju svoju primenu i da je potrebno izvršiti analizu baze i upita koji se najčešće koriste da bi se izabrali odgovarajući indeksi.

LITERATURA

[1] Kurs *Sistemi za upravljanje bazama podataka* na Elektronskom fakultetu u Nišu - https://cs.elfak.ni.ac.rs/nastava/course/view.php?id=57

[2] „Database Management Systems“ - Raghu Ramakrishnan & Johannes Gehrke

[3] „Indexes - SQL Server | Microsoft Docs“ - https://docs.microsoft.com/en-us/sql/relational-databases/indexes/indexes?view=sql-server-ver15

[4] „SQL Server Index Architecture and Design Guide - SQL Server | Microsoft Docs“ - https://docs.microsoft.com/en-us/sql/relational-databases/sql-server-index-design-guide?view=sql-server-ver15

[5] „Columnstore indexes - Data loading guidance - SQL Server | Microsoft Docs“ - https://docs.microsoft.com/en-us/sql/relational-databases/indexes/columnstore-indexes-data-loading-guidance?view=sql-server-2017

[6] „What are Columnstore Indexes ? - Simple Talk“ - https://www.red-gate.com/simple-talk/sql/sql-development/what-are-columnstore-indexes/