



Sistemi za upravljanje bazama podataka

Interna statistika koju SQL Server održava

Mentor:

doc. dr Aleksandar Stanimirović

Student:

Natalija Mitić 1046

Sadržaj

Sadržaj.....	2
Uvod.....	3
Statistika DBMS-a	3
Statistika SQL Server-a.....	4
Histogram.....	4
Prikupljanje statistike – statistike indeksa i kolona	7
Čuvanje statistike	8
Kreiranje statistike	8
Automatsko kreiranje statistike.....	8
Manuelno kreiranje	8
Ažuriranje statistike	9
Automatsko ažuriranje	9
Automatsko asinhrono ažuriranje	10
Manuelno ažuriranje	10
Upotreba statistike	11
Kada kreirati statistiku	13
Predikat upita sadrži više povezanih kolona	13
Upit identifikuje nedostajuću statistiku	14
Kada ažurirati statistiku	15
Filtrirana statistika	16
Inkrementalna statistika	19
Zaključak.....	21
Literatura.....	22

Uvod

Baza podataka, pored podataka koje pamti radi obezbeđivanja traženih informacija korisniku kroz aplikaciju, čuva i dodatne podatke o njihovoj organizaciji. Prikupljeni podaci se organizuju, čineći statistike.

Statistika ima značajnu ulogu prilikom optimizacije upita, jer obezbeđuje optimizatoru upita informacije o broju redova koji se očekuje da bude vraćen kao rezultat upita. Na osnovu toga se izračunava cena različitih planova upita, pa se može izabrati onaj koji najviše štedi vreme i memoriju.

U ovom radu je predstavljena statistika koja se čuva kod SQL Server-a. Prikazano je kada se statistika kreira i ažurira. Takođe, objašnjeno je i ilustrovano na primerima koji se podaci pamte i kako se koriste prilikom optimizacije upita.

Statistika DBMS-a

Bez statističkih podataka sačuvanih u katalogu sistema, optimizatoru bi bilo teško da izvrši optimizaciju. Statistike pružaju optimizatoru informacije o stanju tabela kojima će pristupiti SQL upit koji se optimizuje. Statistika, zapravo, pruža ideje o najboljim načinima za obradu upita i, samim tim, maksimizuje performanse. Zato, bez statistike, optimizatoru bi bilo teško unapred odrediti najbolju strategiju obrade.

Kod različitih DBMS-ova se najčešće čuvaju:

- Informacije o tabelama, uključujući ukupan broj redova, informacije o kompresiji i ukupan broj stranica
- Informacije o kolonama uključujući broj diskretnih vrednosti za kolone i raspon vrednosti vrednosti u koloni
- Informacije o skladištu tabela, uključujući broj aktivnih stranica
- Trenutno stanje indeksa, uključujući da li indeks postoji ili ne, organizaciju indeksa (broj listova i broj nivoa), broj diskretnih vrednosti za ključ i da li je indeks klasterovan
- Informacije o particijama

Optimizator upita koristi statistike da proceni kardinalnost, odnosno broj redova u rezultatu upita i tako napravi visokokvalitetni plan upita. Na primer, u zavisnosti od predikata konkretnog upita, optimizator može da koristi procene kardinalnosti za odabir operatora pretraživanja indeksa umesto operatora skeniranja indeksa koji više koristi resurse, ako to poboljšava performanse upita.

Statistika koju koristi optimizator je skup podataka koji opisuju bazu podataka i objekte u njoj. Kako se objekti u bazi podataka mogu neprestano menjati, statistika se mora redovno ažurirati, tako da tačno opisuje sve objekte baze podataka.

Statistike sadrže metrike o broju i distribuciji podataka u jednoj ili više kolona koje koristi optimizator za izbor plana izvršenja upita.

Statistika nad indeksom se automatski kreira kada se kreira indeks i postoji sve dok indeks postoji. Statistika kolona se kreira ručno ili automatski i može se kreirati, modifikovati i brisati po volji.

Statistički podaci sadrže dve različite vrste informacija o podacima: gustinu i distribuciju. Gustina je vrednost obrnuta broju različitih vrednosti kolone. Distribucija je reprezentacija podataka sadržanih u prvoj koloni statistike.

Statistički podaci čine histogram. Histogram je aproksimacija distribucije podataka za kolonu. On može sa razumnom tačnošću reći da li su podaci iskrivljeni ili ne, što će zauzvrat pomoći optimizatoru upita da shvati prirodu podataka koje sadrži baza podataka.

Histogrami se kod različitih DBMS-ova mogu kreirati na različite načine. Najčešći su:

- Histogram jednake širine - Podeoci su unapred definisani.
- Histogram jednake visine - Svaki podeok na histogramu predstavlja skup vrednosti iz baze podataka, a sam podeok je granična vrednost.
- Histogram najčešćih vrednosti - Sastoji se od liste vrednosti i broja njihovih ponavljanja. Kod ograničenog broja podeoka, u histogram će biti uključene samo najčešće vrednosti.

Histogram obezbeđuje informacije o distribuciji podataka u koloni. Na osnovu toga se računa kardinalnost za vrednost koja se nalazi u predikatu upita. Optimizator upita uzima u obzir sve moguće načine za izvršavanje upita i izabere optimalno rešenje, odnosno određeni plan za izvršenje upita.

Statistika SQL Server-a

Statistike koje se koriste za optimizaciju upita su binarni veliki objekti (BLOB) koji sadrže statističke informacije o raspodeli vrednosti u jednoj ili više kolona tabele ili indeksiranog prikaza. Svaki objekt statistike kreira se nad jednom ili više kolona u tabeli i uključuje histogram koji prikazuje raspodelu vrednosti u prvoj koloni. Objekti statistike nad više kolona čuvaju, takođe, statističke podatke o korelaciji vrednosti među kolonama. Ove korelacione statistike, odnosno gustine, izvode se iz broja različitih vrednosti kolone svih redova.

Histogram

Histogram meri učestalost pojavljivanja za svaku zasebnu vrednost u skupu podataka, odnosno pamti distribuciju podataka kolone. Optimizator upita izračunava histogram na osnovu vrednosti iz prve ključne kolone objekta statistike, birajući vrednosti kolona statističkim uzorkovanjem redova ili vršeći potpuno skeniranje svih redova u tabeli. Ako je histogram kreiran iz uzorkovanog skupa redova, sačuvane zbirne vrednosti za broj redova i broj različitih vrednosti u koloni su procenjeni i ne moraju da budu celi brojevi.

Da bi kreirao histogram, optimizator upita sortira vrednosti kolona, izračunava broj vrednosti koji se podudaraju sa svakom različitom vrednošću kolone i zatim objedinjuje te vrednosti u

najviše 200 neprekidnih podeoka histograma. Svaki podeok histograma uključuje raspon vrednosti kolona do gornje granice koja je jednaka vrednosti podeoka. Raspon uključuje sve moguće vrednosti kolona između graničnih vrednosti, ne uključujući same granične vrednosti. Najniža od sortiranih vrednosti kolone je gornja granična vrednost za prvi podeok histograma.

Na slici 1 je prikazan histogram koji je kreiran nad kolonom *Kolicina* u tabeli *Knjiga*, koja predstavlja broj primeraka knjige. Kolona *RANGE_HI_KEY* predstavlja podeoke na histogramu. *RANGE_ROWS* označava koliko ima vrednosti između te tačke na histogramu i prethodne, ne uključujući te dve vrednosti. *EQ_ROWS* označava koliko postoji redova sa baš tom vrednošću *RANGE_HI_KEY*-a. *DISTINCT_RANGE_ROWS* označava koliko postoji različitih vrednosti u opsegu vrednosti između dva podeoka, odnosno broj različitih vrednosti od ukupnog broja vrednosti iz *RANGE_ROWS*. *AVG_RANGE_ROWS* predstavlja odnos *RANGE_ROWS* i *DISTINCT_RANGE_ROWS*, odnosno koliko prosečno postoji redova za svaku različitu vrednost.

	RANGE_HI_KEY	RANGE_ROWS	EQ_ROWS	DISTINCT_RANGE_ROWS	AVG_RANGE_ROWS
1	1	0	1	0	1
2	2	0	2	0	1
3	3	0	5	0	1
4	4	0	2	0	1
5	5	0	7	0	1
6	6	0	4	0	1
7	7	0	1	0	1
8	8	0	4	0	1
9	9	0	1	0	1
10	10	0	933	0	1
11	11	0	1	0	1
12	13	2	2	1	2
13	14	0	1	0	1
14	15	0	2	0	1
15	17	0	1	0	1
16	25	1	1	1	1
17	36	0	1	0	1
18	56	0	1	0	1
19	62	0	1	0	1
20	63	0	1	0	1
21	65	0	1	0	1

Slika 1 - Histogram

Pored samog histograma, prilikom prikaza statistike, prikazane su još neke vrednosti. Na slici 2 se može videti tabela, čije kolone označavaju ime statistike koja je kreirana, datum kada je kreirana, broj redova tabele, kao i broj redova koji su uzeti kao uzorak, zatim broj podeoka na histogramu, gustinu, prosečan broj bajtova po vrednosti za sve ključne kolone, oznaku da li je indeks tipa *String*, izraz kojim je filtrirana statistika i broj redova koji nisu obuhvaćeni filterom statistike.

	Name	Updated	Rows	Rows Sampled	Steps	Density	Average key length	String Index	Filter Expression	Unfiltered Rows
1	brojKnjiga	Mar 25 2020 5:03PM	976	976	21	0.6666667	4	NO	NULL	976

Slika 2 - Heder

Na slici 3 je prikazana ukupna gustina ($1/\text{broj različitih vrednosti}$), prosečan broj bajtova za pamćenje liste vrednosti kolona za prefiks podeoka i lista kolona za koje su prikazana prethodna dva podatka.

	All density	Average Length	Columns
1	0.04347826	4	Kolicina

Slika 3 – Vektor gustine

Optimizator upita koristi ove vrednosti na različite načine kako bi procenio broj redova koji predstavljaju rezultat upita.

SQL Server kreira histogram iz sortiranog skupa vrednosti kolona u tri koraka:

1. Inicijalizacija histograma: U prvom koraku se obrađuje niz vrednosti od početka sortiranog skupa i prikupljaju se sve dok se ne sakupi do 200 vrednosti za range_high_key, equal_rows, range_rows i distinct_range_rows. Prvi korak se završava ili kada su svi unosi prikupljeni ili kada je pronađeno 200 vrednosti.
2. Skeniranje i spajanje: U ovom koraku se svaka dodatna vrednost iz vodeće kolone statistike obrađuje, po sortiranom redosledu. Svaka sukcesivna vrednost se ili dodaje poslednjem rasponu ili se stvara novi raspon na kraju (ovo je moguće jer su ulazne vrednosti sortirane). Ako se stvori novi raspon, tada se jedan par postojećih, susednih raspona sakuplja u jedinstven raspon. Ovaj par raspona je izabran da bi se minimizovao gubitak informacija. Ova metoda koristi algoritam maksimalne razlike da bi minimizovao broj podeoka na histogramu uz maksimizovanje razlike između graničnih vrednosti. Broj koraka nakon spajanja opsega ostaje 200 tokom ovog koraka.
3. Integracija histograma: U trećem koraku mogu se spojiti više raspona ako se ne izgubi značajna količina informacija. Broj podeoka histograma može biti manji od broja različitih vrednosti, čak i za kolone sa manje od 200 graničnih tačaka. Stoga, čak i ako kolona ima više od 200 jedinstvenih vrednosti, histogram može imati manje od 200 podeoka. Za kolonu koja se sastoji samo od jedinstvenih vrednosti, tada će integrirani histogram imati najmanje tri podeoka.

Vektor gustine

Za kreiranje histograma se, pored podataka o distribuciji, koji su prethodno opisani, koriste i podaci o korelaciji. Statistika korelacije, poznata i kao gustina, sadrži broj različitih redova, odnosno vrednosti u koloni. Ovo pomaže u određivanju broja dupliranih vrednosti između kolona. Statistički podaci o korelaciji se prikupljaju kada je navedeno više od jedne kolone iz iste tabele.

Gustina je informacija o broju duplikata u datoj koloni ili kombinaciji kolona i izračunava se kao $1 / (\text{broj različitih vrednosti})$. Vrednost gustine je između 0.0 i 1.0. Optimizator upita koristi gustinu da bi poboljšao procene kardinalnosti za upite koji sadrže više kolona iz iste tabele ili indeksiranog prikaza. Kako se gustina smanjuje, tako selektivnost neke vrednosti raste. Na primer, u tabeli koja predstavlja knjige, mnoge knjige imaju istog izdavača, ali

svaka knjiga ima jedinstveni identifikacioni broj (ISBN). Indeks nad ISBN-om je selektivniji od indeksa izdavača, jer ISBN ima manju gustinu od izdavača. Velika gustina podrazumeva da su podaci manje jedinstveni, dok mala gustina znači da postoji više jedinstvenih podataka.

Vektor gustine sadrži jednu gustinu za svaki prefiks kolona u objektu statistike. Na primer, ako objekat statistike ima ključne kolone *Naslov*, *Autor* i *Zanr*, gustina se izračunava na svakom od sledećih prefiksa kolona: (*Naslov*), (*Naslov, Autor*), (*Naslov, Autor, Zanr*). Gustina nije dostupna za obrnuti redosled kolona, npr. (*Autor, Naslov*) kao ni za slučaj kada je izostavljena kolona iz sredina (*Naslov, Zanr*).

Kolone bi trebalo da budu redosleda prioriteta s leva na desno. Za kreiranje histograma koristi se samo prva kolona. Sve ostale kolone se koriste za statistiku korelacije među kolonama zvanim gustinama.

Na slici 4 je prikazan vektor gustine za statistiku koja je kreirana nad kolonama *Naslov*, *Autor* i *Zanr* i ona ima tri reda.

	All density	Average Length	Columns
1	0.3333333	4.022244	Naslov
2	0.3333333	8.044489	Naslov, Autor
3	0.3333333	12.091	Naslov, Autor, Zanr

Slika 4 – Vektor gustine za statistiku nad više kolona

Kolona *All density* se koristi za izračunavanje prosečnog broja redova koji će biti vraćeni kao rezultat upita koji sadrži tu kolonu ili kombinaciju kolona. Ono što vektor gustine ne pruža je način da se prepozna pojedinačna selektivnost bilo koje od sekundarnih kolona. Poznate su samo kombinacije zasnovane na levom prefiksu.

Prikupljanje statistike – statistike indeksa i kolona

Kada se u predikat upita u WHERE klauzuli uključi neka kolona, SQL Server kreira objekat statistike. Ovako kreirana statistika je statistika kolone, a može se kreirati i ručno za proizvoljnu kolonu.

SQL Server, takođe, automatski generiše objekat statistike kada se kreira indeks nad tabelom. Na primer, kada je kreirana tabela *Knjiga*, SQL Server je generisao objekat statistike na osnovu kolone *ISBN*. Budući da je kolona definisana kao primarni ključ, SQL Server je automatski kreirao klasterovani indeks nad tom kolonom i generisao pridruženi objekat statistike. Objekat statistike će biti kreiran i ako se indeks kreira ručno.

Dakle, ne postoji suštinska razlika između statistike nad indeksom i statistike nad kolonom. One se kreiraju u različitim trenucima i, osim ako korisnik ne kreira manuelnu statistiku, one se uglavnom ne razlikuju. Statistički podaci o indeksu kreiraju se na osnovu njega. Za indeks kreiran nad postojećim podacima obavlja se potpuno skeniranje tih podataka, što se takođe koristi za kreiranje statistika za indeks. Kod statistika kolona, kod velikih tabela, one se mogu kreirati korišćenjem uzorkovanih podataka, a ne potpunim skeniranjem.

Čuvanje statistike

Statistika se čuva u samoj bazi podataka u nizu internih tabela. Većina detalja o njima se prikazuje pomoću funkcije DBCC SHOW_STATISTICS (slike 1,2 i 3). Sama statistika zauzima vrlo malo prostora. Zaglavlje čini jedan red informacija. Gustina je skup redova sa tri kolone, gde je broj redova jednak broju kolona koje definišu ključne kolone statistike. Zatim, histogram sadrži do 200 redova i nikada ne prelazi taj broj. To znači da za statistiku uopšte nije potrebno mnogo prostora.

Kreiranje statistike

Automatsko kreiranje statistike

Kada je opcija automatskog kreiranja statistike, AUTO_CREATE_STATISTICS uključena, optimizator upita kreira statistiku o pojedinim kolonama koje su navedene u predikatu upita, po potrebi, kako bi poboljšao procene kardinalnosti za plan izvršenja upita. Podrazumevano, opcija je uključena. Ove statistike pojedinačnih kolona kreiraju se nad kolonama koje u postojećem objektu statistike nemaju već kreiran histogram. Opcija AUTO_CREATE_STATISTICS ne određuje da li će se statistika kreirati za indekse. Ova opcija takođe ne generiše filtrirane statistike. Primenjuje se isključivo za statistiku pojedinačnih kolona za celu tabelu.

SQL Server će kreirati statistiku kolone kada se vrši upit nad kolonom koja nije vodeća kolona u indeksu. Na primer, ako imamo upit:

```
select * from Knjiga
where opis='aaa'
go ,
```

pod pretpostavkom da *opis* nije prva kolona bilo kog postojećeg indeksa, SQL Server će kreirati statistiku za tu kolonu. To će omogućiti optimizatoru da izabere prilično dobar plan izvršenja. Ove automatski kreirane statistike će imati naziv obrasca:

_WA_Sys_00003_xxxxx. Komponente imena su WA za državu Vašington u kojoj su živeli i radili originalni stvaraoci SQL Servera, zatim Sys od *System*, zatim broj koji odgovara ID-u kolone, a poslednji deo je heksadecimalna vrednost ID-a objekta tabele.

Manuelno kreiranje

Čak i ako je omogućeno automatsko kreiranje statistike, u nekim okolnostima će možda ipak biti korisno da se ručno kreira statistika za kolonu ili skup kolona. To se može učiniti pomoću naredbe CREATE STATISTICS. Može se čak kreirati statistika i pomoću WHERE klauzule u definiciji koja se naziva filtrirana statistika. Na slici 5 je prikazano kreiranje statistike koje ima naziv *multistat_knjiga_autor_zanr* nad kolonama *Naslov*, *Autor* i *Zanr* u tabeli *Knjiga*.

```
CREATE STATISTICS multistat_knjiga_autor_zanr on Knjiga (Naslov, Autor, Zanr)
GO
```

Slika 5 – Kreiranje statistike nad više kolona

Ažuriranje statistike

Automatsko ažuriranje

Kada je omogućeno automatsko ažuriranje statistike (opcija `AUTO_UPDATE_STATISTICS`), SQL Server će pratiti izmene podataka u kolonama. Kada broj promena dostigne određenu tačku izvršiće se automatsko ažuriranje i to:

- Kada se u tabelu bez ijednog reda doda red
- Kada se 500 redova promeni u tabeli koja ima manje od 500 redova
- Kada se promeni $500 + 20\%$ redova u tabeli koja ima više od 500 redova

Počevši od SQL Servera 2016, on koristi padajući, dinamički prag za ažuriranje statistike koji se prilagođava broju redova u tabeli za veće tabele (više od 500 redova). On se izračunava kao kvadratni koren proizvoda broja 1000 i trenutne kardinalnosti tabele. Na primer, ako tabela sadrži 2 miliona redova, izračunava se $\sqrt{(1000 * 2000000)} = 44721.359$. Ovom novinom, statistika nad velikim tabelama će se češće ažurirati.

Pod promenom se podrazumeva dodavanje, brisanje ili modifikacija redova. Automatsko ažuriranje ne pokreće celokupno skeniranje, već se uzima uzorak određenog procenta redova i očitava se samo neki interno izračunati deo redova. Obično je ovo dovoljno tačno i nema toliko veliki uticaj na performanse kao potpuno skeniranje. Ako je korisniku potrebno potpuno skeniranje, za to je potrebno zakazati posao. To je slučaj kod tabela u kojima su podaci izrazito iskrivljeni ili veoma nestabilni.

Kad god se indeks nad tabelom obnovi, statistika se ažurira sa potpunim skeniranjem i svi planovi za izvršavanje upita koji koriste te statistike postaju nevažeći. Međutim, ako su indeksi samo reorganizovani, ne dolazi do ažuriranja statistika i nijedan plan nije nevalidan.

Kada je opcija za automatsko ažuriranje statistike uključena, optimizator upita proverava zastarele statistike pre kompajliranja upita i pre izvršenja keširanog plana upita. Statistički podaci mogu postati zastareli kada operacije umetanja, ažuriranja, brisanja ili spajanja tabela menjaju distribuciju podataka u tabeli. Pre kompajliranja upita, optimizator upita koristi kolone, tabele i indeksirane prikaze u predikatu upita kako bi odredio koja je statistika verovatno zastarela. Pre izvršenja keširanog plana izvršenja upita, mehanizam za proveru podataka (eng. Database engine) proverava da li plan upita koristi ažurne statistike. Optimizator upita određuje kada je statistika zastarela tako što broji broj modifikacija podataka od poslednjeg ažuriranja statistike i upoređuje broj modifikacija sa pragom.

Opcija `AUTO_UPDATE_STATISTICS` odnosi se na objekte statistike kreirane za indekse, za pojedinačne kolone navedene u predikatima upita i statistike kreirane naredbom `CREATE STATISTICS`. Treba voditi računa da, kada se ažurira statistika, statistike više kolona ili filtrirane statistike se ne ažuriraju automatski. Ažuriranje se mora obaviti ručno.

Kada se podaci u tabeli promene, indeksi se automatski ažuriraju. Kako se sve više i više podataka menja, statistike postaju sve manje i manje tačne. Kvalitet planova za izvršenje

upita stvorenih iz neažurnih statistika je loš. To je razlog zašto je dobro uključiti opciju za automatsko ažuriranje.

Automatsko asinhrono ažuriranje

Opcija za asinhrono ažuriranje statistike, `AUTO_UPDATE_STATISTICS_ASYNC`, određuje da li će optimizator upita koristiti sinhrono ili asinhrono ažuriranje statistike. Ova opcija se odnosi na objekte statistike kreirane za indekse, pojedinačne kolone iz predikata upita i statistike kreirane naredbom `CREATE STATISTICS`.

Kod sinhronog ažuriranja statistika standardni postupak je sledeći: Kada se izvrši upit, optimizator ispituje stanje osnovnih statistika. Ako je potrebno ažurirati ih, SQL mehanizam (eng. SQL engine) ih ažurira i kreira plan izvršenja na osnovu novih statistika i poništava sve druge planove koji koriste staru statistiku. Zatim izvršava upit koristeći novu statistiku i novi plan izvršenja. Dakle, kada su statistike zastarele, optimizator upita čeka ažurirane statistike pre nego što kompajlira i izvrši upit.

Da bi se izbegao ovaj proces koji uglavnom duže traje, asinhrona opcija izvršava upit odmah koristeći stare statistike i stari plan izvršenja. Zatim ažurira statistiku i poništava sve prethodne planove koji su je koristili. Kompenzacija je u tome što se upit može izvršiti bez čekanja, ali će koristiti staru statistiku, pa se može desiti da se odabere manje optimalan plan za izvršenje upita.

Neka se, na primer, koristi sinhrono ažuriranje statistike kada se izvode operacije koje menjaju distribuciju podataka, kao što su odsecanje tabele ili vršenje ažuriranja velikog procenta redova. Ako se ne ažurira statistika nakon završetka operacije, upotreba sinhronog ažuriranja statistika će osigurati da su statistike ažurirane pre izvršenja upita nad izmenjenim podacima.

Korišćenje asinhronog ažuriranja statistike je pogodno za postizanje predvidljivijih vremena odgovora na upit za sledeće scenarije:

- Aplikacija često izvršava isti upit, slične upite ili slične planove upita. Vreme odgovora na upit može biti predvidljivije kod asinhronog ažuriranja statistika nego kod sinhronog, jer optimizator upita može izvršavati nove upite bez čekanja na ažurne statistike. Na taj način se izbegava odlaganje nekih upita.
- Aplikacija je iskusila istek vremena klijenskog zahteva prouzrokovanog jednim ili više upita koji čekaju ažurirane statistike. U nekim slučajevima čekanje na sinhronu statistiku može prouzrokovati pad aplikacija sa agresivnim intervalima isteka vremena.

Manuelno ažuriranje

Statistika se može ručno ažurirati da bi se poboljšao plan izvršenja upita i performanse na osnovu zahteva. Može se koristiti naredba `UPDATE STATISTICS` ili procedura `Sp_Update` za ažuriranje statistika SQL Servera. U nastavku slede načini ažuriranja statistika.

- UPDATE STATISTICS za sve statistike u objektu
`update statistics Knjiga`
- UPDATE STATISTICS za specifične statistike
`update statistics Knjiga incstat`
- UPDATE STATISTICS sa *full scan* opcijom
Full scan opcija se koristi za skeniranje svih redova tabele. U prethodnim primerima nije naveden ovaj parametar, stoga SQL Server automatski odlučuje da li mu je potrebno potpuno skeniranje ili ne. Sledeći upit vrši potpuno skeniranje i ažurira statistike za određene statistike u navedenom objektu.
`update statistics Knjiga incstat with fullscan`
- UPDATE STATISTICS sa uzorkom
Sa *SAMPLE* klauzulom se može specificirati procenat ili broj redova za ažuriranje statistike.
`update statistics Knjiga brojKnjiga with sample 10 percent`
- UPDATE STATISTICS svih statistika kolona jednog objekta
`update statistics Knjiga with fullscan , columns`
Ova opcija se može koristiti kod kreiranja indeksa, kako bi se ažurirale sve statistike u objektu, jer se tada samo statistike nad indeksom automatski ažuriraju.

Kod manualnog ažuriranja treba obratiti pažnju na neke specifičnosti:

- Treba voditi računa da se *SAMPLE* opcija koristi samo kod specifičnih zahteva. U suprotnom možda će biti uzeta manja količina uzorka, što možda neće biti pogodno da optimizator upita odabere odgovarajući plan.
- Ne bi trebalo onemogućiti automatsko ažuriranje statistike čak i ako korisnik sam redovno ažurira statistiku. Automatsko ažuriranje statistike omogućava SQL serveru da automatski ažurira statistiku prema unapred definisanom pragu.
- Ažuriranje statistika pomoću *FULL SCAN*-a može trajati duže za veliki skup podataka. Zato bi trebalo pažljivo odabrati period kada će se ažurirati statistika.

Ažuriranje pomoću sp_updatestats

Procedura *sp_updatestats* se koristi za ažuriranje svih statistika u bazi podataka, tako što prolazi kroz svaku statistiku objekata i vrši traženo ažuriranje. Statistike se ažuriraju ukoliko je to potrebno. Proverava se da li su bilo kakvi redovi modifikovani u tabeli na koju se statistika odnosi. Ako je modifikovan jedan ili više redova, ažuriraće se statistika. Za velike baze podataka će možda biti potrebno duže vreme i sistemski resursi, jer se vrši provera svake statistike u objektu.

Problem sa korišćenjem ove naredbe može biti tome što se ažuriraju samo statistike za koje se smatra da su neispavane, bez obzira na to šta korisnik želi.

Upotreba statistike

Statistike koristi optimizator upita da bi pronašao prihvatljiv plan izvršenja za upite koji se odnose na određene kolone. Uz dobru statistiku, optimizator može donositi informisane odluke o stvarima kao što su: kojim redosledom treba da se vrši spoj tabela; kakav spoj treba

koristiti; koji indeks treba koristiti; da li treba koristiti pretragu ili skeniranje tog indeksa itd. Tačnos odluke u ovim oblastima zavise od tačnosti statistike. Dobre statistike, čak, mogu napraviti razliku između upita koji se izvršava za manje od sekunde ili koji traje ceo jedan sat. Optimizator upita treba da zna koliko će redova biti vraćeno kao rezultat upita i na osnovu toga bira odgovarajući plan. Ova procena broja redova se dobija iz statistika.

Neka se izvrši upit:

```
select * from Knjiga
where Kolicina=4;
```

Na slici 6 se može videti da je procenjeni broj redova koji čine rezultat identičan stvarnom broju redova koji je vraćen.

Actual Number of Rows	2
Number of Rows Read	16
Estimated Number of Rows	2
Estimated Row Size	5265 B
Estimated Data Size	10 KB

Slika 6 – Kardinalnost upita za Kolicina=4

Optimizator upita koristi histogram za kolonu *Kolicina* kako bi izračunao ovu procenu. Pošto vrednost 4 postoji kao podeok histograma, procenjena vrednost je tačna (slika 7).

	RANGE_HI_KEY	RANGE_ROWS	EQ_ROWS	DISTINCT_RANGE_ROWS	AVG_RANGE_ROWS
1	1	0	13	0	1
2	2	0	2	0	1
3	3	0	5	0	1
4	4	0	2	0	1
5	5	0	7	0	1
6	6	0	4	0	1

Slika 7 – Histogram (prvih 6 podeoka)

Međutim ukoliko se za količinu postavi npr. 61 procena je približna, ali nije tačna. Procenjen broj redova je 1, a stvaran je 0 (slika 8).

Actual Number of Rows	0
Number of Rows Read	16
Estimated Number of Rows	1
Estimated Row Size	5265 B
Estimated Data Size	5265 B

Slika 8 - Kardinalnost upita za Kolicina=61

Pošto 61 nije podeok na histogramu, optimizator upita koristi vrednost AVG_RANGE_ROWS (slika 9). Podeoci predstavljaju gornju granicu opsega, zato je vrednost uzeta iz reda sa vrednošću 62.

17	36	0	1	0	1
18	50	0	1	0	1
19	62	1	1	1	1
20	63	0	1	0	1
-	-	-	-	-	-

Slika 9 – Histogram (17-20. podeok)

Kada kreirati statistiku

Kao što je spomenuto, optimizator upita sam kreira statistiku u sledećim slučajevima:

1. Kada se kreira indeks. Ove statistike kreiraju se na ključnim kolonama indeksa. Ako je indeks filtriran, optimizator upita stvara filtriranu statistiku na istom podskupu redova koji su navedeni za filtrirani indeks.
2. Nad pojedinačnim kolonama u predikatima upita ako je uključena opcija za automatsko kreiranje statistike.

Za većinu upita, ove dve metode kreiranja statistika osiguravaju visokokvalitetni plan upita. Dodatne statistike mogu, u nekim slučajevima, poboljšati planove upita. Ove dodatne statistike mogu obuhvatiti statističke korelacije koje optimizator upita ne vodi prilikom kreiranja statistika za indekse ili pojedinačne kolone. Aplikacija može imati dodatne statističke korelacije u podacima tabela koje bi, ako se uzmu u obzir u objektu statistike, mogle omogućiti optimizatoru upita da poboljša planove upita. Na primer, filtrirana statistika na podskupu redova podataka ili statistika više kolona u predikatu upita mogu poboljšati plan upita.

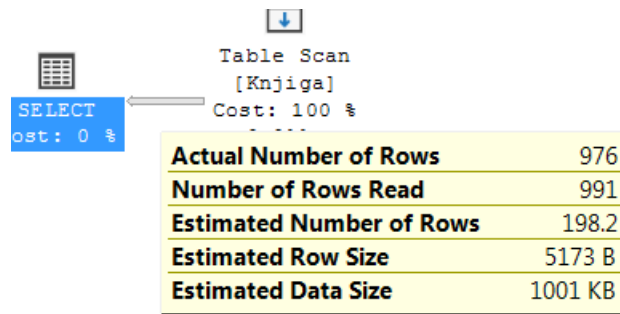
Predikat upita sadrži više povezanih kolona

Kada predikat upita sadrži više kolona koje međusobno imaju veze i zavisnosti, statistika nad više kolona može poboljšati plan izvršenja upita. Statistički podaci o više kolona sadrže korelacijske statistike između kolona, zvane gustine, koje nisu dostupne u statistikama o jednoj koloni. Gustine mogu poboljšati procene kardinalnosti kada rezultati upita zavise od zavisnosti podataka između više kolona. Ako su kolone već u istom indeksu, objekt statistike više kolona već postoje i nije ih potrebno ručno kreirati. Za održavanje indeksa je potrebno više sistemskih resursa u odnosu na objekat statistike. Zato, ako aplikacija ne zahteva indeks nad više kolona, može se uštedeti na resursima kreiranjem objekta statistike bez kreiranja indeksa. Prilikom kreiranja statistika nad više kolona redosled kolona u definiciji objekta statistike utiče na efikasnost gustina za procenu kardinalnosti. Objekat statistike čuva gustinu za svaki prefiks ključnih kolona u definiciji objekta statistike. Na slici 10 je prikazan vektor gustine za statistiku nad kolonama *Naslov* i *Autor*. Postoje 4 različita naslova i 5 različitih parova naslov-autor (gustina = 1/ broj različitih vrednosti).

	All density	Average Length	Columns
1	0.25	4.020182	Naslov
2	0.2	8.042381	Naslov, Autor

Slika 10 – Vektor gustine

Procenjeni broj redova koji se vraćaju kao rezultat je 198.2 (slika 11). Ukupno postoji 991 red, pa je $991 * 0.2 = 198.2$, što znači da je korišćena gustina kako bi se dobila procena.



Slika 11 – Procenjena kardinalnost

Upit identifikuje nedostajuću statistiku

Ako greška ili neki drugi događaj sprečavaju optimizator upita da kreira statistiku, on kreira plan upita bez korišćenja statistike. Optimizator upita označava statistiku kao nedostajuću i pokušava da je obnovi sledeći put kada se upit izvrši. Nedostajuće statistike su označene kao upozorenja kada se plan izvršenja upita grafički prikazuje pomoću *SQL Server Management Studio-a*. Svakako, optimizator izračunava procenu kardinalnosti iako statistika ne postoji.

Ako se izvrši upit:

```
declare @e int
set @e=4
select * from Knjiga
where Kolicina=@e;
```

procena kardinalnosti se neće poklopiti sa stvarnim brojem redova (slika 12). U ovom slučaju optimizator upita ne može pristupiti histogramu, jer vrednost lokalne promenljive e nije dostupna optimizatoru upita sve do izvršenja. Zato se primenjuje podrazumevana procena kardinalnosti za operator = : Procenjeni broj = Ukupan broj redova * Gustina. Bez histograma se podrazumeva uniformna distribucija podataka. Na slici 13 se vidi da je gustina 0.04166667, a pošto tabela ima 991 red, množenjem se dobija 41.2917. Pošto se histogramu ne može pristupiti, za bilo koju vrednost parametra e će procena kardinalnosti biti ista.

Kod primene operatora *BETWEEN*, procena kardinalnosti se dobija kao 9% od ukupnog broja redova u tabeli, dok se za operatore $<$, $>$, $<=$, $>=$ dobija kao 30% od ukupnog broja redova.

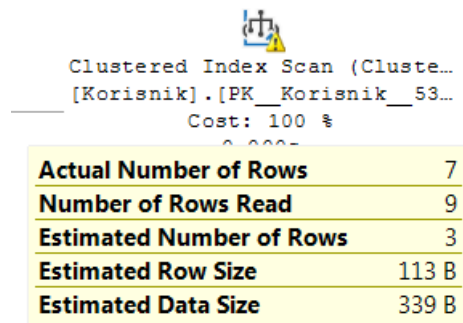
Actual Number of Rows	2
Number of Rows Read	19
Estimated Number of Rows	41.2917
Estimated Row Size	5265 B
Estimated Data Size	212 KB

Slika 12 – Procenjena kardinalnost za $e=4$

	All density	Average Length	Columns
1	0.04166667	4	Kolicina

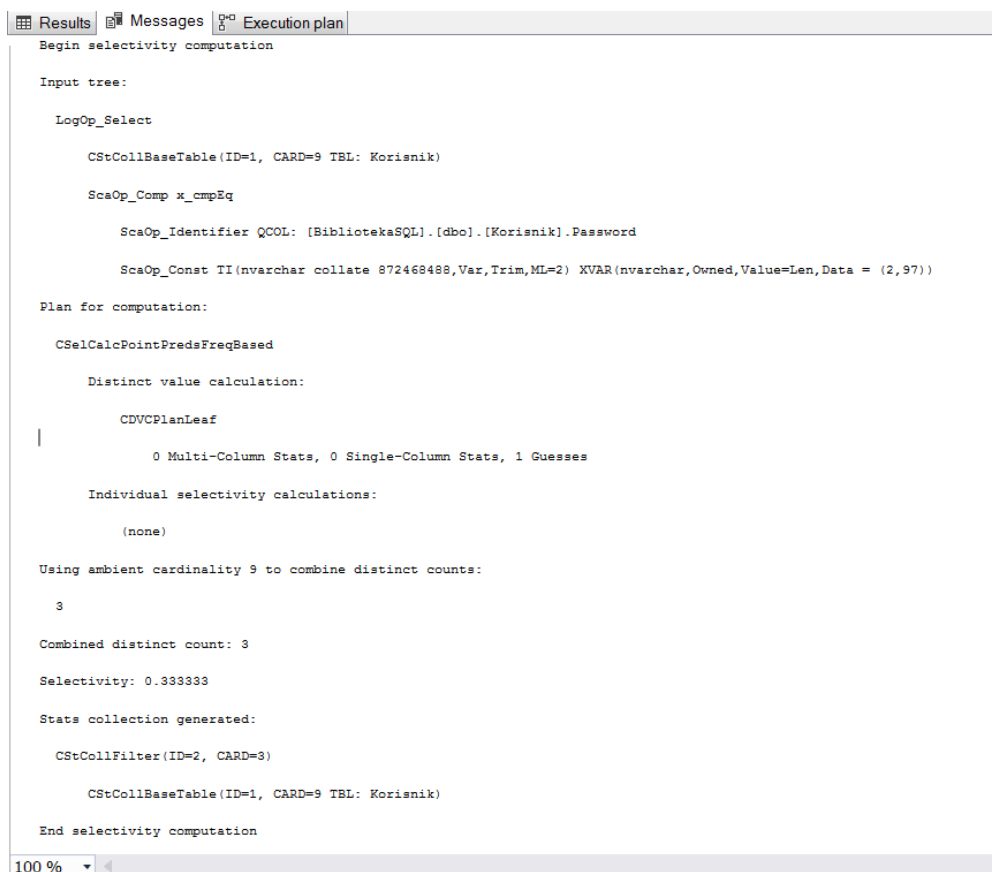
Slika 13 – Vektor gustine statistike nad kolonom Kolicina

Kada optimizator upita uopšte ne može da koristi statistiku, on mora da izvrši nagađanje kako bi dobio neku procenu (slika 14). Računa se koren broja redova tabele i on predstavlja procenjenu kardinalnost. Na slici 15 je prikazan način izvršenja upita, gde se vidi da se ne koristi statistika i da je izvršeno nagađanje.



Clustered Index Scan (Cluste... [Korisnik].[PK_Korisnik_53... Cost: 100 %	
Actual Number of Rows	7
Number of Rows Read	9
Estimated Number of Rows	3
Estimated Row Size	113 B
Estimated Data Size	339 B

Slika 14 – Procenjena kardinalnost bez korišćenja statistike



Results	Messages	Execution plan
Begin selectivity computation		
Input tree:		
LogOp_Select		
CStCollBaseTable(ID=1, CARD=9 TBL: Korisnik)		
ScaOp_Comp x_cmpEq		
ScaOp_Identifier QCOL: [BibliotekaSQL].[dbo].[Korisnik].Password		
ScaOp_Const TI(nvarchar collate 872468488,Var,Trim,ML=2) XVAR(nvarchar,Owned,Value=Len,Data = (2,97))		
Plan for computation:		
CSELCalcPointPredsFreqBased		
Distinct value calculation:		
CDVCPPlanLeaf		
0 Multi-Column Stats, 0 Single-Column Stats, 1 Guesses		
Individual selectivity calculations:		
(none)		
Using ambient cardinality 9 to combine distinct counts:		
3		
Combined distinct count: 3		
Selectivity: 0.333333		
Stats collection generated:		
CStCollFilter(ID=2, CARD=3)		
CStCollBaseTable(ID=1, CARD=9 TBL: Korisnik)		
End selectivity computation		

Slika 15 – Računanje kardinalnosti bez korišćenja statistike

Kada ažurirati statistiku

Ažuriranje statistike osigurava da se upiti kompajliraju sa ažurnim statistikama. Međutim, ažuriranje statistika uzrokuje rekompiliranje upita. Zato se preporučuje da se statistike ne ažuriraju prečesto, jer postoji kompromis performansi između poboljšanja planova izvršenja

upita i vremena potrebnog za rekompajliranje upita. Specifični kompromisi zavise od same aplikacije.

Operacije umetanja se dešavaju u rastućem ili opadajućem redosledu u ključnim kolonama

Statistika rastućih i opadajućih ključnih kolona, poput vremenskih oznaka u realnom vremenu, može zahtevati češća ažuriranja nego što ih vrši optimizator upita. Broj dodatih redova je možda premali da bi pokrenuo ažuriranje statistike. Ako statistika nije ažurna i upiti izvlače podatke iz poslednje dodanih redova, trenutna statistika neće imati procene kardinalnosti za ove nove vrednosti. To može rezultovati netačnim procenama kardinalnosti i sporim izvršenjem upita.

Operacije održavanja

Korisno je ažurirati statistiku nakon izvođenja postupaka održavanja koji menjaju distribuciju podataka, poput odsecanja tabele ili izvođenja istovremenog umetanja velikog procenta redova. Ovo može izbeći buduća kašnjenja u obradi upita, dok upiti čekaju automatsko ažuriranje statistika. Operacije poput rebildovanja, defragmentacije ili reorganizacije indeksa ne menjaju distribuciju podataka. Zbog toga nije potrebno ažurirati statistiku. Optimizator upita ažurira statistiku kada se rebilduje indeks nad tabelom ili pogledom, međutim ovo ažuriranje statistike je nusprodukt ponovnog kreiranja indeksa.

Filtrirana statistika

U SQL Serveru je moguće kreiranje filtrirane statistike. One su izuzetno korisne u radu sa particionisanim podacima ili podacima koji su iskrivljeni zbog širokog opsega podataka ili puno NULL vrednosti. Takođe, mogu biti korisne u veoma velikim tabelama.

Prilikom kreiranja statistike može se koristiti WHERE klauzula. Ona određuje izraz za odabir podskupa redova koje treba uključiti prilikom kreiranja objekta statistike. Statistika koja je ovako kreirana naziva se filtrirana statistika. Ako je potreban jedan kriterijum za sve SQL upite kada se preuzimaju podaci iz te tabele, može se kreirati filtrirana statistika nad tom tabelom koja će doprineti tačnosti plana izvršenja upita.

Filtrirana statistika može poboljšati performanse upita za upite koji biraju podatke iz dobro definisanih podskupova podataka. Dobro dizajnirana filtrirana statistika može poboljšati plan izvršavanja upita u poređenju sa statistikama nad celim tabelama.

Histogram je limitiran na 200 podeoka, bez obzira na to koliko redova ima tabela. Kod velikih tabela će većina vrednosti biti negde između podeoka histograma, pa će se koristiti prosečne vrednosti (AVG_RANGE_ROWS) za procenu broja redova koji čine rezultat upita. Ovakva procena može biti prilično loša, naročito ako su podaci dosta iskrivljeni. Filtrirana statistika omogućava “zumiranje”, odnosno kreiranje statistika za više skupova podataka, tako da se dobije bolje procena, jer se histogrami odnose na uži skup podataka.

Filtrirana statistika je korisna kada upiti biraju iz podskupa redova, a taj podskup redova ima jedinstvenu distribuciju podataka. Na primer, svaki podatak iz tabele *Kategorija*, baze podataka za prodavnicu biciklističke opreme, pripada jednoj od četiri kategorije: bicikl,

komponente, odeća i dodaci. Svaka od kategorija ima različitu distribuciju podataka za težinu: vrednosti težine bicikala se kreću npr. od 13.77 do 30, težine komponenata se kreću od 2.12 do 1050 sa nekim NULL vrednostima, težine odeće su NULL, a težine dodatne opreme su takođe NULL. Za ovaj primer, filtrirana statistika o svim težinama bicikala će pružiti tačniju statistiku optimizatoru upita i može poboljšati kvalitet plana izvršenja upita u odnosu na statistiku cele tabele.

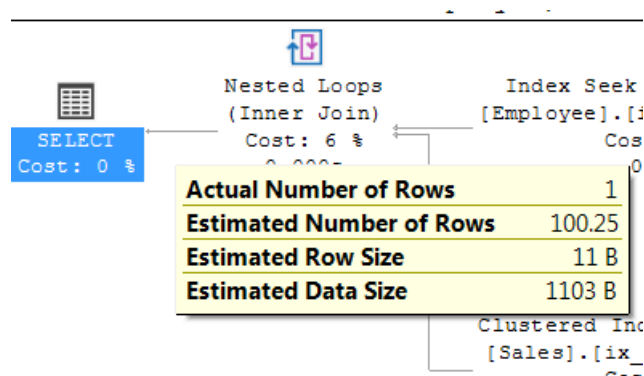
Mnogo puta posoje indeksi i njihove statistike, kao i statistike kolona, ali optimizator još uvek nije u mogućnosti da napravi tačnu procenu, jer ne razume međusobni odnos podataka. Kad se spoje dve tabele, procenjena kardinalnost može biti veća ili manja od stvarnog broja traženih redova. Pošto je procenjena kardinalnost netačna, SQL Server ne može tačno da odabere najbolji plan.

Na primer, imamo dve tabele: *Employee* i *Sales*. Tabela *Employee* pamti id-eve i imena zaposlenih, a tabela *Sales* id radnika koji je izvršio prodaju, količinu knjiga koju je prodao i datum svake prodaje. Pretpostavimo da je Jelena novi radnik i izvršila je tek jednu prodaju, dok je Milan radnik sa dužim stažom i obavio je 200 prodaja.

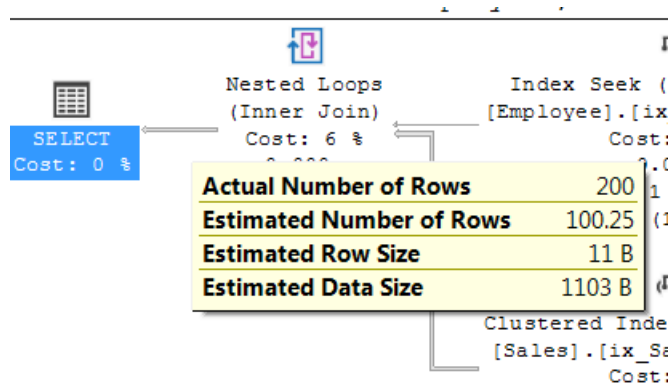
Ako se izvrši sledeći upit:

```
SELECT salesamount
FROM employee,sales
WHERE employee.empid = sales.empid
AND name = 'Jelena'
GO ,
```

rezultat će vratiti 1 red, jer Jelena ima jednu prodaju, a ako se u predikatu navede Milan, rezultat će sadržati 200 redova. Na slikama 16 i 17 je prikazan plan za izvršenje upita kada su u predikatu za ime navedeni Jelena i Milan respektivno. U oba slučaja je procenjeni broj redova 100.25, što je prilično daleko od stvarnog broja redova koji je vraćen u oba sličaja. Ovaj broj je dobijen na sledeći način: tabela *Employee* ima 4 reda, a tabela *Sales* 401. Kada se podele ova dva broja dobija se 100.25.



Slika 16 – Procenjena kardinalnost za name=Jelena

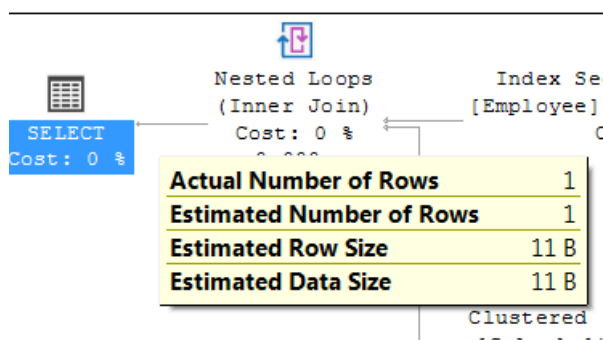


Slika 17 – Procenjena kardinalnost za name=Milan

Filtrirana statistika se može koristiti za rešavanje ove greške. Sledeći upit je korišćen za kreiranje filtrirane statistike:

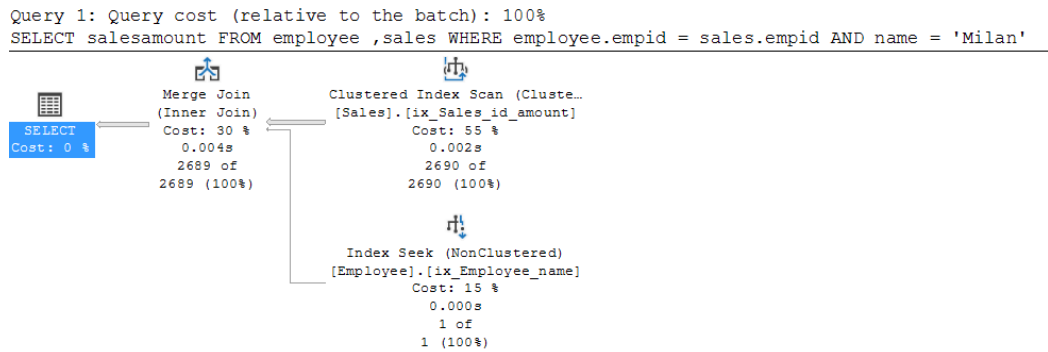
```
CREATE STATISTICS Employee_stats_EmpID ON Employee (EmpID)
WHERE name = 'Jelena'
GO
```

Takođe je kreirana statistika i za name = Milan. Nakon kreiranja statistika, ponovo je izvršen prethodni upit. Na slici 18 se može videti da je ovog puta procenjeni broj redova tačan.

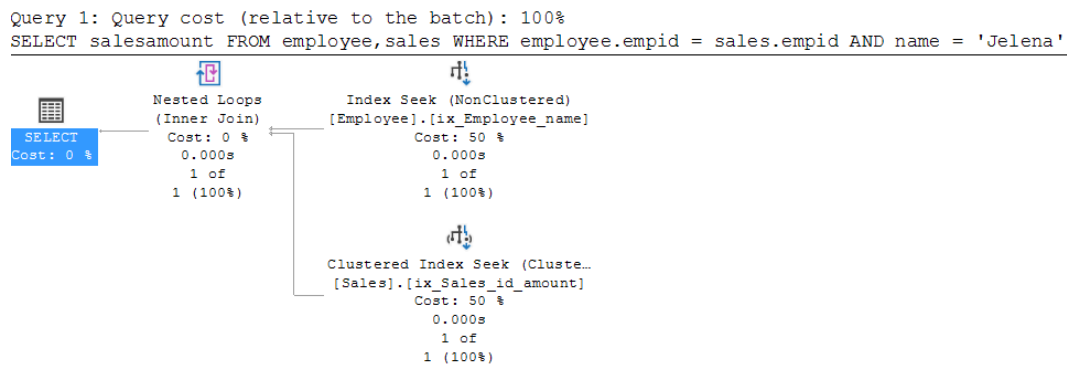


Slika 18 – Procenjena kardinalnost korišćenjem filtrirane statistike

Na osnovu filtriranih statistika optimizator upita može izabrati pogodniji plan izvršenja. Na slikama 19 i 20 je prikazano izvršenje kada je navedeno ime Milan i Jelena respektivno kod kojih se može uočiti da se prikupljanje podataka iz tabele *Sales* vrši različito u oba slučaja, jer se u prvom slučaju vraća 2690 redova, a u drugom 1.



Slika 19 – Izvršenja upita za name=Milan



Slika 20 - Izvršenja upita za name=Jelena

Dobre i loše strane filtrirane statistike

Treba voditi računa da, ako je kreirana statistika nad nekom kolonom, tip kolone se ne može promeniti dok statistika postoji. S druge strane filtrirane statistike su “lightweight” objekti, pa se zato mogu lakše održavati nego filtrirani indeksi nad velikim tabelama. Kreiranje i održavanje indeksa nije lako izvodljivo i sporo je. Kolona sa težinom bicikla iz gore spomenutog primera je dobar kandidat za filtriranu statistiku, ali nije nužno i dobar kandidat za filtrirani indeks, ako je broj pregleda težine relativno mali. Dobitak performansi za pretrage koje filtrirani indeks pruža možda ne prevazilaze dodatne troškove održavanja i skladištenja filtriranog indeksa u bazi podataka. Međutim, mana filtriranih statistika nad velikim tabelama je to što mogu brzo postati neažurne. Ako se automatski ažurira statistika, prag ažuriranja će se odnositi na celu tabelu. Ako se podaci promene samo u delu tabele koji je obuhvaćen filtriranom statistikom, može se lako desiti da prag nije dostignut, a podaci su promenjeni tako da statistika više uopšte ne oslikava njihovu distribuciju.

Inkrementalna statistika

SQL Server zahteva skeniranje cele table radi ažuriranja statistike i to izaziva probleme kod velikih tabela. Ovo takođe važi i za tabele sa particijama.

Kad je opcija INCREMENTAL za kreiranje statistike uključena, statistike su kreirane prema statistikama particija. Kada je isključena, stablo statistike se odbacuje i SQL Server ponovo izračunava statistiku. Podrazumevano je isključena.

Inkrementalna statistika nije podržana za sledeće tipove statistika:

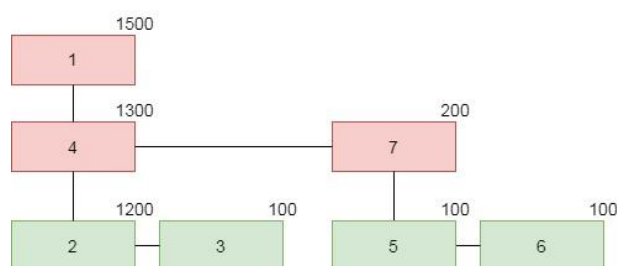
- Statistika kreirana nad indeksima koji nisu poravnati sa particijama za osnovnu tabelu
- Statistika kreirana nad bazama podataka samo za čitanje
- Statistika kreirana nad filtriranim indeksima
- Statistika kreirana nad prikazima
- Statistika kreirana nad internim tabelama
- Statistika kreirana pomoću prostornih indeksa ili XML indeksa

Kada se nove particije dodaju u veliku tabelu, statistiku treba ažurirati tako da uključi nove particije. Međutim, vreme potrebno za skeniranje čitave tabele (opcija FULLSCAN ili SAMPLE) može biti prilično dugo. Takođe, skeniranje čitave tabele nije neophodno, jer će biti potrebni samo statistički podaci o novim particijama. Inkrementalna opcija kreira i čuva statistike nad particijama, a kada se ažurira, ažurira samo statistiku onih particija kojima su potrebne nove statistike.

Ako distribucija podataka ostaje ista za sve prethodne particije, tada je potrebno da *SQL Server* zna promenjenu distribuciju podataka za novu ili novoažuriranu particiju. Kada je opcija INCREMENTAL uključena, novokreirane statistike će koristiti inkrementalnu statistiku za particionisane tabele.

Svaka particija može sadržati do 200 podeoka za formiranje histograma. Međutim, *SQL Server* ne koristi ovu statistiku nivoa particija za procenu kardinalnosti. Glavna statistika, koja se ažurira iz statistike na nivou particija, je statistika koju će koristiti *SQL Server*.

Tabela *korisnik* je podeljena na osnovu id-a na 4 particije i kreirana je inkrementalna statistika. Na slici 21 je prikazano kako je kreirano stablo statistike. Zeleni listovi stabla se odnose na particije statistika, dok su crvenim obeležene stranice koje su dobijene spajanjem po dva čvora. Čvor 1 predstavlja glavnu statistiku. Kada se ažurira statistika neke particije, promene se propagiraju kroz stablo statistike sve do glavne statistike.



Slika 21 – Stablo inkrementalne statistike

Nakon ubacivanja 1000 redova u tabelu, gde svi spadaju u poslednju particiju, izvršeno je ažuriranje statistike samo te particije:

```
update statistics [korisnik] (incstat)
with resample1 on partitions (4)
go
```

```
SQL Server parse and compile time:
    CPU time = 0 ms, elapsed time = 0 ms.

SQL Server Execution Times:
    CPU time = 16 ms,  elapsed time = 9 ms.
SQL Server parse and compile time:
    CPU time = 0 ms, elapsed time = 0 ms.

Completion time: 2020-04-12T11:48:19.2391818+02:00
```

Slika 22 – Vreme ažuriranja statistike nad 4. particijom

```
SQL Server parse and compile time:
    CPU time = 0 ms, elapsed time = 0 ms.

SQL Server Execution Times:
    CPU time = 16 ms,  elapsed time = 19 ms.
SQL Server parse and compile time:
    CPU time = 0 ms, elapsed time = 0 ms.

Completion time: 2020-04-12T11:47:20.7158345+02:00
```

Slika 23 – Vreme ažuriranja statistike potpunim skeniranjem

Slika 22 prikazuje vreme ažuriranja kada se ažurira samo poslednja particija, a slika 23 kad se koristi potpuno skeniranje. Za tabelu koja sadrži 1500 redova razlika je 10ms. Ako je tabela veća i vremenska razlika će biti značajnija.

Bez primene inkrementalne statistike, potrebno je da se promeni 20% podataka cele tabele da bi se izvršilo ažuriranje statistike. Međutim, sa inkrementalnom statistikom je isti prag primenjiv na jednu particiju. To znači da ako se particija suoči sa promenom 20% podataka, promena će pokrenuti i ažuriranje statistika na nivou particije koje će, kada se spoji sa konačnom statistikom, pružiti bolje performanse. Tačnije, prag je broj redova/broj particija * 20%.

Zaključak

Bez statističkih podataka, optimizator upita bi teško mogao da izvrši optimizaciju. Ove statistike pružaju optimizatoru informacije o stanju tabela kojima će se pristupiti prilikom izvršenja SQL upita. Zbog njihove važnosti, preporučuje se da se uključe opcije automatskog kreiranja i ažuriranja statistika. Dodatno se mogu ručno kreirati statistike kako bi se procena broja redova koje vraća upit povećala, što dovodi do izbora boljeg plana za izvršenje upita. Cilj bolje procene je brže izvršenje upita, što rezultuje brži odgovor korisniku i poboljšano zadovoljstvo.

¹ Opcija resample se mora koristiti, jer sve statistike u novoj strukturi stabala moraju biti usklađene s istim uzorkom, kako bi se kasnije spojile s glavnom statistikom.

Literatura

- [1] – Managing optimizer statistics, Oracle help center
https://docs.oracle.com/cd/B19306_01/server.102/b14211/stats.htm#g49431
- [2] – Optimizer statistics, MySQL documentation
<https://dev.mysql.com/doc/refman/8.0/en/optimizer-statistics.html>
- [3] - Global Statistics vs. Histograms with DBMS_STATS Package, Christoph Gächter
https://www.akadia.com/services/ora_gather_statistics.html
- [4] – Understanding SQL Server Statistics, SQL Consulting
<https://www.sqlconsulting.com/archives/understanding-sql-server-statistics/>
- [5] – Statistics, Microsoft SQL Server documentation
<https://docs.microsoft.com/en-us/sql/relational-databases/statistics/statistics?view=sql-server-ver15>
- [6] – SQL Statistics Basic, Redgate Hub, Robert Sheldon
<https://www.red-gate.com/simple-talk/sql/performance/sql-server-statistics-basics/>
- [7] - SQL Server Statistics and how to perform Update Statistics in SQL, SQLShack
<https://www.sqlshack.com/sql-server-statistics-and-how-to-perform-update-statistics-in-sql/>
- [8] - Introducing SQL Server Incremental Statistics for Partitioned Tables, SQLShack
<https://www.sqlshack.com/introducing-sql-server-incremental-statistics-for-partitioned-tables/>
- [9] – New incremental statistics, dbi Services
<https://blog.dbi-services.com/sql-server-2014-new-incremental-statistics/>
- [10] – What is filtered statistics, SQL Authority, Dave Pinal
<https://blog.sqlauthority.com/2015/04/07/sql-server-what-is-filtered-statistics/>
- [11] – Filtered Statistics Follow-up, Brent Ozar
<https://www.brentozar.com/archive/2016/12/filtered-statistics-follow/>
- [12] - Filtered Stats and CE Model Variation in SQL Server, SQLShack
<https://www.sqlshack.com/filtered-stats-and-ce-model-variation/>