

**Univerzitet u Nišu**

**Elektronski fakultet**

**Katedra za računarstvo**

**Seminarski rad**

**Interna statistika kod SQL Server baze podataka**

**Sistemi za upravljanje bazama podataka**

|  |  |  |
| --- | --- | --- |
| **Mentor**: |  | **Student**: |
| Dr Aleksandar Stanimirović |  | Miloš Veljanovski 1559 |

**Niš, avgust 2023.**

**Sadržaj**

[1. Uvod 3](#_Toc143357371)

[1.1. SQL Server 3](#_Toc143357372)

[1.2. Obrada upita 3](#_Toc143357373)

[2. Osnovi statistike kod SQL Servera 4](#_Toc143357374)

[2.1. Prvi koraci 5](#_Toc143357375)

[2.2. Pribavljanje informacija o statističkom objektu 6](#_Toc143357376)

[2.3. Statistički objekti zasnovani na kolonama 7](#_Toc143357377)

[2.4. Statistički objekti zasnovani na indeksima 7](#_Toc143357378)

[3. Histogrami statistika 8](#_Toc143357379)

[3.1. Vektor gustine 10](#_Toc143357380)

[3.2. Primeri 10](#_Toc143357381)

[4. Opcije ze statistiku 15](#_Toc143357382)

[4.1. Opcija AUTO\_CREATE\_STATISTICS 15](#_Toc143357383)

[4.2. Opcija AUTO\_UPDATE\_STATISTICS 15](#_Toc143357384)

[4.3. Opcija AUTO\_UPDATE\_STATISTICS\_ASYNC 16](#_Toc143357385)

[4.4. Opcija AUTO\_DROP 18](#_Toc143357386)

[4.5. Opcija INCREMENTAL 18](#_Toc143357387)

[6. Literatura 27](#_Toc143357388)

[7. Listing 28](#_Toc143357389)

# 1. Uvod

## **1.1. SQL Server**

SQL Server je sistem za upravljanje relacionom bazom podataka razvijen od strane Microsoft-a. To je moćna i fleksibilna platforma koja je široko rasprostranjena u poslovanju i organizacijama svih obima. Neke od glavnih odlika SQL Servera su čuvanje i upravljanje podacima, skalabilnost, optimizacija performansi, sigurnost, poslovna inteligencija i analitika, razvojni alati.

SQL Server pruža otpornu platformu za čuvanje i upravljanje podacima, koja uključuje podršku za tabele, poglede (*eng. views*), procedure, okidače (*triggers*) i mnoge druge objekte u bazi. Podržava skaliranje sa manjih okruženja sa jednim serverom, na velike multi-server konfiguracije. Ovo se postiže tehnologijama poput klasterovanja i replikacije. SQL Server takođe uključuje i brojne alate za optimizaciju performansi baze podataka, gde spadaju planovi izvršenja upita, indeksiranje i statistike. Zaštita podataka se ogleda u robusnim bezbednosnim funkcionalnostima, kao što su enkripcija, upravljanje pristupom zasnovanom na ulogama, revizija. SQL Server implementira i podršku za poslovnu analitiku kroz alate kao što su servisi SQL Servera za analizu (*SQL Server Analysis Services - SSAS*), servisi za izveštaje (*SQL Server Reporting Services - SSRS*) i integracioni servisi (*SQL Server Integration Services - SSIS)*. Od razvojnih alata SQL Server nudi moćne alate i API-je razvojnim timovima, kao što su SQL Server Menadžment Studio (SSMS), SQL Server Alat za podatke (*SQL Server Data Tools - SSDT*), i podršku za popularne programske jezike T-SQL, C# i Java.

SQL Server dostupan je u više edicija, kao što su *Enterprise, Standard* i *Express.* Enterprise nudi najveći skup mogućnosti i alata, dok je Express edicija besplatna i lakša verzija dizajnirana za razvojna okruženja manjih obima. SQL Server je moćna i svestrana platforma koja pomaže organizacijama svih veličina u efikasnoj analizi podataka.

## **1.2. Obrada upita**

Obrada upita je ključna komponenta sistema za upravljanje bazama podataka, koji obuhvata prevođenje i izvršavanje korisničkih upita nad bazom. Primarni cilj obrade upita je transformacija upita visokog nivoa pisanih na deklarativnom jeziku (na primer, SQL) u plan izvršenja niskog nivoa. Ovaj plan sastoji se od sekvence operacija koje se mogu efikasno izvršiti nad bazom.

Obrada upita uglavnom uključuje sledeće korake:

1. Parsiranje – Ulazni upit se parsira da bi se obezbedila njegova skladnost sa sintaksom jezika upita.
2. Semantička analiza – Upit se analizira da bi se osiguralo da je semantički ispravan i smislen. Ovaj korak vrši provere kao što su ispravnost imena tabela i kolona i pravilna upotreba operatora.
3. Optimizacija – Optimizator upita generiše različite planove izvršenja za isti upit i bira najefikasniji u pogledu vremena izvršenja i iskorišćenih resursa. Optimizacione tehnike uključuju optimizaciju baziranu na ceni, optimizaciju baziranu na pravilima i heurističku optimizaciju.
4. Izvršenje - Odabrani plan izvršenja se pokreće nad bazom, pritom generišući rezultate samog upita.

Za nas će u ovom radu koji se bavi statistikom ponajviše od interesa biti treći korak (optimizacije upita) ponajviše zahvaljujući činjenici da je SELECT izraz ne-proceduralni, tj. ne zadaje konkretne korake koje server treba da iskoristi da bi nabavio tražene podatke. Ovo znači da server mora da analizira izraz da bi odlučio koji je najefikasniji način za izvlačenje potrebnih podataka. Komponenta koja radi ovo naziva se Optimizator upita (*Query Optimizer*). Ulaz na ovu komponentu sastoji se od upita, šeme baze (definicije tabela i indeksa), i statistike baze. Izlaz iz optimizatora je plan izvršenja upita, koji se ponekad naziva i plan upita, ili plan izvršenja.

A diagram of a diagram

Description automatically generated

Slika 1. Optimizator upita

Naravno, glavni ciljevi optimizacije SQL upita jesu poboljšanje performansi smanjenjem vremena odziva, smanjenje vremena trajanja izvođenja na procesoru (vremenska razlika između zahteva korisnika za podacima i dobijanja odgovora treba biti minimalna radi boljeg korisničkog iskustva), kao i povećanje propusnosti (broj resursa kojima treba pristupiti kako bi se dobili svi potrebni podaci treba biti minimiziran).

# 2. Osnovi statistike kod SQL Servera

Statistike za optimizaciju upita su binarni veliki objekti (*BLOB*-ovi) koji sadrže statističke informacije o raspodeli vrednosti u jednoj ili više kolona tabele ili indeksiranog pogleda. SQL Serverov optimizator upita koristi statistike kako bi odredio dobar plan izvršenja za naš SQL upit. Ne moramo znati o njima da bismo izvršili upite, ali što bolje razumemo ove statistike, to bolje možemo optimizovati svoje upite i rešiti probleme sa performansama. SQL Serverov optimizator upita koristi statistike prilikom kreiranja plana upita. Statistike pružaju informacije o distribuciji vrednosti kolona među učestvujućim redovima, pomažući optimizatoru da bolje proceni broj redova ili kardinalnost rezultata upita.

Statistike pružaju optimizatoru naznake o najboljim načinima obrade upita i time maksimiziraju performanse. Na primer, statistika bi mogla navesti optimizator da izabere pretragu (*seek*) indeksa umesto skeniranja (*scan*) indeksa, što je potencijalno zahtevnija operacija po pitanju resursa. Bez statistika, bilo bi teško optimizatoru da unapred odredi najbolju strategiju obrade.

Optimizator upita se uglavnom brine o generisanju i ažuriranju statistika distribucije, kreirajući ih na određenim kolonama unutar tabele ili indeksiranog pogleda (*view*-a). Svaki skup statistika čuva se kao objekat na nivou tabele koji prati distribuciju vrednosti u prvoj koloni u bilo kom datom skupu. Ako je objekat statistike definisan na više kolona, objekat takođe čuva detalje o korelaciji vrednosti u kolonama. Ove statistike korelacije, nazvane gustine, baziraju se na broju različitih redova vrednosti kolona.

Iako gustine igraju važnu ulogu u optimizaciji upita, ovaj rad će se pretežno baviti statistikama distribucije pošto im se pristupa preko statističkih objekata. SQL Server takođe pruža statistike u obliku histograma koji pokazuje kako su podaci raspoređeni unutar kolone.

Kada optimizator upita obavlja svoj posao i vaši upiti se izvršavaju kako očekujete, ne morate previše brinuti o histogramima, statističkim objektima ili distribucijskim statistikama uopšte. S druge strane, ako pokušavate da maksimizirate performanse upita ili pratite probleme vezane za performanse, može biti vredno istražiti kako se statistika koristi i generiše i možda u skladu s tim izvršiti prilagođenja.

Međutim, pre nego što to možete da uradite, morate imati osnovno razumevanje o tome kako se statistika implementira u SQL Serveru, kako da pratite njihove povezane histograme i kako da shvatite smisao informacija, o čemu ćemo se baviti u ovom radu.

## **2.1. Prvi koraci**

Kao i sa mnogim oblastima SQL Servera, statistiku distribucije može biti lakše razumeti ako ih vidite na delu, umesto da jednostavno čitate o njima apstraktno. Da bi pomogao u tom procesu, ovaj rad uključuje nekoliko primera koji ilustruju kako se statistike distribucije generišu i kako pristupiti informacijama o njima.

Za ove primere, koristio sam sledeću T-SQL skriptu da kreiram tabelu *AWSales* i popunim je podacima iz uzorak baze podataka zvane *AdventureWorks2019*:

IF OBJECT\_ID (N'dbo.AWSales', N'U') IS NOT NULL

DROP TABLE dbo.AWSales;

GO

CREATE TABLE dbo.AWSales

(

OrderID INT PRIMARY KEY IDENTITY(101, 1),

SalesOrderID INT NOT NULL,

OrderDetailID INT NOT NULL,

ProductID INT NOT NULL,

LineTotal MONEY NOT NULL

);

INSERT INTO dbo.AWSales

(SalesOrderID, OrderDetailID, ProductID, LineTotal)

SELECT SalesOrderID, SalesOrderDetailID, ProductID, LineTotal

FROM AdventureWorks2019.Sales.SalesOrderDetail

WHERE SalesOrderDetailID < 60000;

Primer 1. Kreiranje AWSales tabele

## **2.2. Pribavljanje informacija o statističkom objektu**

SQL Server automatski generiše statistički objekat kada kreirate indeks na tabeli ili indeksirani pogled. Na primer, prilikom kreiranja tabele *AWSales*, SQL Server je generisao statistički objekat zasnovan na koloni *OrderID*. Budući da je kolona definisana kao primarni ključ, SQL Server je automatski kreirao klasterizovani indeks na toj koloni i generisao pripadajući statistički objekat.

SQL Server pruža sledeća dva sistemska pogleda za dobijanje okvirnih informacija o statističkim objektima povezanim sa tabelom ili indeksiranim pogledom:

* sys.stats: Vraća ID tabele ili indeksiranog pogleda, ime i ID statističkog objekta, izraz za filtriranje (ako je primenljivo), i niz bitskih zastavica (*flags*) koje ukazuju na informacije kao što su da li je statistički objekat automatski generisan ili korisnički definisan.
* sys.stats\_columns: Vraća ID nadređenog (roditeljskog) objekta, ID statističkog objekta, redosledni položaj svake kolone uključene u statistički objekat i ID kolone, kako je naveden unutar *sys.columns*.

Ova dva pogleda možemo spojiti na osnovu ID-eva objekta i statistike kako bismo dobili određene tipove informacija. Takođe, možemo dodati pogled *sys.columns* kako bismo dobili stvarna imena kolona, kao što je prikazano u sledećem primeru:

SELECT s.stats\_id StatsID,

s.name StatsName,

sc.stats\_column\_id StatsColID,

c.name ColumnName

FROM sys.stats s

INNER JOIN sys.stats\_columns sc

ON s.object\_id = sc.object\_id AND s.stats\_id = sc.stats\_id

INNER JOIN sys.columns c

ON sc.object\_id = c.object\_id AND sc.column\_id = c.column\_id

WHERE OBJECT\_NAME(s.object\_id) = 'awsales'

ORDER BY s.stats\_id, sc.column\_id;

Primer 2. Dobijanje informacija o statističkim objektima pomoću sistemskih pogleda

U ovom slučaju, jedino što me zanima su ID tabele i ID statistike, zajedno sa kolonama na kojima se statistike temelje. Kao što pokazuju sledeći rezultati, tabela trenutno uključuje samo jedan statistički objekat i taj objekat se temelji samo na jednoj koloni, OrderID-u.

|  |  |  |  |
| --- | --- | --- | --- |
| StatsID | StatsName | StatsColID | ColumnName |
| 1 | PK\_\_AWSales\_\_C3905BAF9C6C0B09 | 1 | OrderID |

Kada se statistički objekat generiše kao rezultat kreiranja indeksa, objekat dobija ime indeksa. Pored toga, da je statistički objekat bio zasnovan na više kolona, *SELECT* naredba bi vratila red za svaku kolonu.

## **2.3. Statistički objekti zasnovani na kolonama**

SQL Server takođe generiše statistički objekat kada uključimo kolonu u uslov upita, kao što je *WHERE* klauzula. Na primer, pretpostavimo da postavljamo upit za tabelu *AWSales* na osnovu vrednosti *ProductID*:

SELECT \*

FROM AWSales

WHERE ProductID = 898

OPTION(RECOMPILE);

Primer 3. Uključivanje uslova u upit pomoću *WHERE* klauzule

U ovom primeru, dodao sam nagoveštaj (*hint*) upitu zvani *RECOMPILE* kako bih primorao optimizator da odbaci postojeći plan upita i time obezbedim sveže rezultate svaki put kada pokrenenem *SELECT* naredbu i pregledam plan izvršenja, o čemu ćemo uskoro razgovarati.

Na mom sistemu, ovaj upit je vratio devet redova. Zatim sam ponovo izvršio istu *SELECT* naredbu iz Primera 2 kako bih video ažuriranu listu statističkih objekata. *SELECT* naredba je ista kao i u prethodnom odeljku, samo sada imamo novi statistički objekat, kao što je prikazano u sledećim rezultatima:

|  |  |  |  |
| --- | --- | --- | --- |
| StatsID | StatsName | StatsColID | ColumnName |
| 1 | PK\_\_AWSales\_\_C3905BAF9C6C0B09 | 1 | OrderID |
| 2 | \_WA\_Sys\_00000004\_39AD8A7F | 1 | ProductID |

SQL Server je automatski kreirao statistički objekat *\_WA\_Sys\_00000004\_39AD8A7F* na osnovu kolone *ProductID*. U ovom slučaju, ime automatski generisanog objekta ima prefiks \_WA, što je dvoslovni kod za saveznu državu Vašington u Sjedinjenim Američkim Državama. Da bi SQL Server generisao objekte bazirane na kolonama, opcija baze podataka *AUTO\_CREATE\_STATISTICS* mora biti omogućena, što podrazumevano i jeste slučaj.

## **2.4. Statistički objekti zasnovani na indeksima**

Kao što smo ranije diskutovali, kada smo definisali primarni ključ našoj tabeli, SQL Server je generisao statistički objekat zasnovan na klasterizovanom indeksnom ključu (kolona OrderID). SQL Server će takođe generisati statistički objekat ako ručno kreiramo indeks, kao što je prikazano u sledećoj skripti:

IF EXISTS (SELECT name FROM sys.indexes WHERE name = N'ixOrders')

DROP INDEX ixOrders ON dbo.AWSales;

GO

CREATE INDEX ixOrders ON dbo.AWSales(SalesOrderID, OrderDetailID);

Primer 4. Kreiranje indeksa u *AWSales* tabeli

U ovom slučaju, kreirali smo neklasterizovani indeks *ixOrders* na kolonama *SalesOrderID* i *OrderDetailID*. Ako bismo sada pokrenuli upit za sistemske prikaze *sys.stats* i *sys.stats\_columns*, koristeći istu *SELECT* naredbu kao i pre, otkrili bismo da je treći statistički objekat dodat u tabelu *AWSales*, kao što je prikazano u sledećim rezultatima.

|  |  |  |  |
| --- | --- | --- | --- |
| StatsID | StatsName | StatsColID | ColumnName |
| 1 | PK\_\_AWSales\_\_C3905BAF9C6C0B09 | 1 | OrderID |
| 2 | \_WA\_Sys\_00000004\_39AD8A7F | 1 | ProductID |
| 3 | ixOrders | 1 | SalesOrderID |
| 3 | ixOrders | 2 | OrderDetailID |

Rezultati uključuju dva reda za novi statistički objekat, po jedan za svaku kolonu u indeksu *ixOrders*. Primetimo da su vrednosti *StatsColID* inkrementirane za svaku kolonu, ali ime i ID statističkog objekta su isti u svakom redu, zajedno sa imenom objekta koje je isto kao ime indeksa.

# 3. Histogrami statistika

Histogram meri učestalost pojavljivanja za svaku različitu vrednost u skupu podataka. Optimizator upita izračunava histogram na vrednostima kolona u prvoj koloni koja predstavlja ključ statističkog objekta, birajući vrednosti kolona statističkim uzorkovanjem redova ili izvršavajući potpuno skeniranje svih redova u tabeli ili pogledu. Ako je histogram kreiran iz uzorkovanog skupa redova, sačuvane ukupne vrednosti za broj redova i broj različitih vrednosti su procene i ne moraju biti celi brojevi.

Da bi kreirao histogram, Optimizator upita sortira vrednosti kolone, izračunava broj vrednosti koje se podudaraju sa svakom različitom vrednošću kolone, a zatim agregira vrednosti kolona u maksimalno 200 kontinuiranih koraka histograma. Svaki korak histograma uključuje opseg vrednosti kolone, praćen gornjom granicom vrednosti kolone. Opseg obuhvata sve moguće vrednosti kolone između granica, isključujući same granične vrednosti. Najniža sortirana vrednost kolone je gornja granična vrednost za prvi korak histograma.

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

1. Inicijalizacija histograma: U prvom koraku se obrađuje niz vrednosti počevši od početka sortiranog skupa i prikuplja se do 200 vrednosti *range\_high\_key*, *equal\_rows*, *range\_rows* i *distinct\_range\_rows* (*range\_rows* i *distinct\_range\_rows* su uvek nula tokom ovog koraka). Prvi korak se završava ili kada se iscrpe svi ulazni podaci ili kada se pronađe 200 vrednosti.
2. Skeniranje sa spajanjem koraka: Svaka dodatna vrednost iz vodeće kolone ključa statistike se obrađuje u drugom koraku, redosledom sortiranja; svaka sledeća vrednost se ili dodaje poslednjem opsegu ili se kreira novi opseg na kraju (ovo je moguće jer su ulazne vrednosti sortirane). Ako se kreira novi opseg, tada se jedan par postojećih, susednih opsega spaja u jedan opseg. Ovaj par opsega se bira kako bi se smanjio gubitak informacija. Ovaj metod koristi algoritam maksimalne razlike kako bi se minimizovao broj koraka u histogramu dok se maksimizira razlika između graničnih vrednosti. Broj koraka nakon spajanja opsega ostaje 200 tokom ovog koraka.
3. Konsolidacija histograma: U trećem koraku, više opsega može biti spojeno ako se ne izgubi značajan deo informacija. Broj koraka 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 koraka. Za kolonu koja se sastoji samo od jedinstvenih vrednosti, konsolidovani histogram će imati najmanje tri koraka.

Sledeći dijagram prikazuje histogram sa šest koraka. Oblast levo od prve gornje granične vrednosti predstavlja prvi korak.

A graph with lines and text

Description automatically generated with medium confidence

Slika 2. Karakteristične vrednosti u histogramu

Za svaki korak navedenog histograma gore važi:

* Podebljana linija predstavlja gornju graničnu vrednost (*range\_high\_key*) i broj puta koliko se pojavljuje (*equal\_rows*).
* Neprozirna površina levo od range\_high\_key predstavlja opseg vrednosti kolone i prosečan broj puta koliko se svaka vrednost kolone pojavljuje (*average\_range\_rows*). Prosečan broj redova opsega za prvi korak histograma je uvek 0.
* Tačkaste linije predstavljaju uzorkovane vrednosti koje se koriste za procenu ukupnog broja različitih vrednosti u opsegu (*distinct\_range\_rows*) i ukupnog broja vrednosti u opsegu (*range\_rows*). Optimizator upita koristi *range\_rows* i *distinct\_range\_rows* kako bi izračunao *average\_range\_rows* i ne čuva uzorkovane vrednosti.

## **3.1. Vektor gustine**

Gustina je informacija o broju duplikata u datoj koloni ili kombinaciji kolona i izračunava se kao 1/(*broj različitih vrednosti*). Optimizator upita koristi gustine kako bi poboljšao procene kardinalnosti za upite koji vraćaju više kolona iz iste tabele ili indeksiranog pogleda. Kako gustina opada, selektivnost vrednosti se povećava. Na primer, u tabeli koja predstavlja automobile, mnogi automobili imaju istog proizvođača, ali svaki automobil ima jedinstveni broj identifikacije vozila (*VIN*). Indeks na VIN je selektivniji od indeksa na proizvođača, jer VIN ima nižu gustinu u poređenju sa proizvođačem.

## **3.2. Primeri**

Do sada smo samo videli kako se dodaju statistički objekti. Iako su sistemski pogledi *sys.stats* i *sys.stats\_columns* korisni za dobijanje okvirnih (*high level*) informacija, oni nam ne pružaju uvid u stvarne statistike distribucije. Za to moramo koristiti *DBCC SHOW\_STATISTICS* naredbu.

Prema podrazumevanoj postavci, *DBCC SHOW\_STATISTICS* naredba vraća sledeće tri vrste informacija:

* Zaglavlje: Ime statističkog objekta, datum i vreme poslednjeg ažuriranja, broj redova u učestvujućim kolonama, stvaran broj uzorkovanih redova, broj koraka unutar histograma i druge informacije.
* Gustine: Vektor gustine indeksiranih kolona na osnovu formule 1/#\_distinct\_values (broj različitih vrednosti). Broj različitih vrednosti pomaže u određivanju selektivnosti vrednosti kolone. Što je više jedinstvenih vrednosti, to je veća selektivnost i efikasniji indeks. Broj redova u ovom odeljku zavisi od broja kolona koje učestvuju.
* Histogram: Distribucija vrednosti po podacima kolone, uvećana u koracima na osnovu broja redova, do 200 koraka po statističkom objektu.

Najmanje što moramo napisati kada pokrećemo *DBCC SHOW\_STATISTICS* naredbu je ciljna tabela ili indeksirani pogled i ciljni indeks, kolona ili statistički objekat. Na primer, sledeća *DBCC SHOW\_STATISTICS* naredba specificira tabelu *AWSales* i statistički objekat *ixOrders*:

DBCC SHOW\_STATISTICS (AWSales, ixOrders);

Primer 5. Korišćenje *DBCC SHOW STATISTICS* naredbe

Naredba *DBCC SHOW\_STATISTICS* podržava i nekoliko drugih opcija, kao što je mogućnost da rezultati uključuju samo histogram, ali za sada ćemo se držati najosnovnijeg oblika naredbe, koji vraća rezultate prikazane na sledećoj slici.

A screenshot of a computer

Description automatically generated

Slika 3. Rezultat izvršavanja *DBCC SHOW STATISTICS* naredbe

Prvi odeljak pruža informacije o zaglavlju, drugi odeljak prikazuje gustine, a treći odeljak nam daje histogram.

U ovom slučaju, histogram je podeljen na 184 koraka, na osnovu opsega podataka u prvoj indeksiranoj koloni, *SalesOrderID*. Svaki korak je naveden u svojem redu u histogramu i predstavlja opseg vrednosti. Na primer, vrednost u koloni *RANGE\_HI\_KEY* u redu 7 je 44133. Ova vrednost predstavlja najvišu vrednost u opsegu podataka koji uključuje sve vrednosti od 44101 (vrednost u redu 6 + 1) do 44113.

Histogram takođe uključuje kolonu *RANGE\_ROWS*, koja pruža broj redova unutar opsega, isključujući redove koji su povezani sa vrednošću *RANGE\_HI\_KEY* (gornjim krajem opsega). U ovom slučaju, u opsegu ima 343 reda, isključujući gornju vrednost opsega. Međutim, vrednost *EQ\_ROWS* označava broj redova koji su povezani sa vrednošću gornjeg kraja. To znači da 17 redova ima vrednost *SalesOrderID*-a 44133.

Kolona *DISTINCT\_RANGE\_ROWS* prikazuje broj jedinstvenih vrednosti unutar trenutnog opsega, ponovo isključujući gornju vrednost opsega. Kao rezultat toga, opseg vrednosti od 44101 do 44132 uključuje 32 jedinstvene vrednosti.

Konačno, kolona *AVG\_RANGE\_ROWS* prikazuje prosečan broj redova za svaku različitu vrednost, prema formuli *RANGE\_ROWS*/*DISTINCT\_RANGE\_ROWS*, ponovo isključujući redove gornjeg kraja. Ovo nam daje prosečno više od 10 redova za svaku različitu vrednost unutar opsega.

Optimizator upita koristi ove vrednosti na različite načine kako bi procenio broj redova koji će biti u rezultatima. Na primer, pretpostavimo da pokrećemo sledeću *SELECT* naredbu, koja ograničava skup rezultata na redove sa vrednošću *SalesOrderID*-a 44133 (red 7 u našem histogramu).

SELECT \*

FROM AWSales

WHERE SalesOrderID = 44133

OPTION(RECOMPILE);

Primer 6. Upit za analizu histograma za graničnu vrednost

Sada pogledajmo plan izvršenja koji se koristi za upit. Sledeća slika prikazuje detalje za *SELECT* čvor plana izvršenja, kako je izgledalo u mojoj lokalnoj instalaciji SQL Server Management Studio-a (SSMS).

A screenshot of a computer

Description automatically generated

Slika 4. Procenjeni plan izvršenja za vrednost koja jeste svoj red

Primetimo da iznos procenjenog broja redova (*Estimated Number of Rows Per Execution*) prikazuje 17. Ovaj broj tačno odgovara vrednosti *EQ\_ROWS* u histogramu statistika za red 7. Ovo je takođe broj redova koje je upit vratio.

Sada pretpostavimo da navedemo vrednost *SalesOrderID*-a od 44134 u našem upitu:

SELECT \*

FROM AWSales

WHERE SalesOrderID = 44134

OPTION(RECOMPILE);

Primer 7. Upit za analizu histograma za ne-graničnu vrednost

Ovoga puta, upit vraća samo jedan red, ali plan izvršenja pokazuje nešto drugo što nije 1, kao što je prikazano na sledećoj slici.

A screenshot of a computer screen

Description automatically generated

Slika 5. Procenjeni plan izvršenja za vrednost koja nije svoj red

Zato što smo naveli vrednost *SalesOrderID*-a koja se nalazi u sledećem opsegu, i nije njen sopstveni red, plan izvršenja koristi vrednost *AVG\_RANGE\_ROWS* iz reda 8, koja je u ovom slučaju 1.493507 (optimizator zaokružuje vrednost na 1.49351), što predstavlja prosečan broj redova po različitim vrednostima u tom opsegu.

Iako to nije previše važno kada se vraća jedan red, to bi moglo napraviti razliku kada se vraća više redova iz velikog skupa podataka, što bi nas navelo da razmislimo o kreiranju ili ažuriranju statistika.

Pogledajmo još jedan primer. Ovoga puta, specificiraćemo opseg vrednosti *SalesOrderID*-a:

SELECT \*

FROM AWSales

WHERE SalesOrderID < 43861

OPTION(RECOMPILE);

Primer 8. Upit za analizu histograma za opseg vrednosti

Ovaj upit vraća 634 reda, što su svi redovi sa vrednošću *SalesOrderID*-a manjom od 43861 (red 3 u histogramu). Izvršni plan takođe dolazi sa procenjenom vrednošću od 634 reda, kao što je prikazano na sledećoj slici.

A screenshot of a computer screen

Description automatically generated

Slika 6. Procenjeni plan izvršenja za opseg vrednosti

Da bi došao do broja 634, optimizator je sabrao statistike iz redova 1, 2 i 3, čime je dobio sledeću jednačinu:

12 + 282 + 28 + 312 = 634

Kao što ovaj poslednji primer pokazuje, optimizator upita može koristiti statistiku distribucije na razne načine da proceni broj redova koji se očekuju u rezultatima. Međutim, imajmo na umu da logika može postati mnogo složenija nego što je prikazano ovde, u zavisnosti od toga kako se pišu upiti nad podacima. Suština je da statistike distribucije igraju ključnu ulogu u računanju plana upita i poznavanje načina pristupa tim statistikama je ključno kako bismo ih razumeli.

# 4. Opcije ze statistiku

Postoje opcije koje utiču na to kada i kako se statistike kreiraju i ažuriraju. Ove opcije su konfigurabilne samo na nivou baze podataka.

## **4.1. Opcija AUTO\_CREATE\_STATISTICS**

Kada je automatska opcija kreiranja statistika, *AUTO\_CREATE\_STATISTICS*, uključena, Optimizator upita kreira statistiku o pojedinačnim kolonama u predikatu upita, po potrebi, kako bi poboljšao procene kardinalnosti za plan upita. Ove statistike pojedinačnih kolona kreiraju se na kolonama koje već nemaju histogram u postojećem statističkom objektu.

Opcija *AUTO\_CREATE\_STATISTICS* ne određuje da li će se statistika kreirati za indekse. Ova opcija takođe ne generiše filtrirane statistike. Ona se striktno odnosi na statistike pojedinačnih kolona za celu tabelu. Kao što je ranije pomenuto u ovom radu, kada Optimizator upita kreira statistike kao rezultat korišćenja opcije *AUTO\_CREATE\_STATISTICS*, naziv statistika počinje sa *\_WA*.

## **4.2. Opcija AUTO\_UPDATE\_STATISTICS**

Kada je automatska opcija ažuriranja statistika, *AUTO\_UPDATE\_STATISTICS*, uključena, Optimizator upita utvrđuje kada bi statistike mogle biti zastarele, a zatim ih ažurira kada ih upit koristi. Ova akcija se takođe naziva rekompilacija statistike. Statistike postaju zastarele nakon što izmene operacija umetanja, ažuriranja, brisanja ili spajanja promene distribuciju podataka u tabeli ili indeksiranom pogledu. Optimizator upita utvrđuje kada statistike mogu biti zastarele brojanjem broja izmena redova od poslednjeg ažuriranja statistika i upoređivanjem broja izmena redova sa pragom. Prag se bazira na kardinalnosti tabele, koja se može definisati kao broj redova u tabeli ili indeksiranom pogledu.

Obeležavanje statistika kao zastarelih na osnovu izmena redova se dešava čak i kada je opcija *AUTO\_UPDATE\_STATISTICS* isključena. Kada je ova opcija isključena, statistike se ne ažuriraju, čak i kada su obeležene kao zastarele. Planovi će i dalje nastaviti da koriste zastarele statističke objekte. Postavljanje opcije *AUTO\_UPDATE\_STATISTICS* na isključeno može uzrokovati suboptimalne planove upita i smanjenje performansi upita, zato se uvek preporučuje postavljanje ove opcije na uključeno.

## **4.3. Opcija AUTO\_UPDATE\_STATISTICS\_ASYNC**

Opcija asinhronog ažuriranja statistika, *AUTO\_UPDATE\_STATISTICS\_ASYNC*, određuje da li optimizator upita koristi sinhrona ili asinhrona ažuriranja statistike. Podrazumevano, opcija asinhronog ažuriranja statistika je isključena, i optimizator upita ažurira statistike sinhrono. Ova opcija se primenjuje na statističke objekte kreirane za indekse, pojedinačne kolone u predikatima upita i statistike kreirane pomoću *CREATE STATISTICS* naredbe.

Da bismo postavili opciju asinhronog ažuriranja statistika u SQL Server Management Studiju, na stranici *Options* (opcije) prozora *Properties* (svojstva) baze podataka, obe opcije *Auto Update Statistics* (Automatsko ažuriranje statistika) i *Auto Update Statistics Asynchronously* (Asinhrono ažuriranje statistika) treba postaviti na *True* (tačno). Na sledećoj slici je prikazan izgled tog prozora.

A screenshot of a computer

Description automatically generated

Slika 7. Podešavanje statističkih opcija kroz dijalog svojstava baze podataka

Ažuriranja statistika mogu biti ili sinhrona (podrazumevano) ili asinhrona.

* Sa sinhronim ažuriranjem statistika, upiti se uvek kompajliraju i izvršavaju sa ažuriranim statistikama. Kada su statistike zastarele, optimizator upita čeka na ažurirane statistike pre nego što kompajlira i izvršava upit.
* Sa asinhronim ažuriranjem statistika, upiti se kompajliraju sa postojećim statistikama čak i ako su postojeće statistike zastarele. Optimizator upita može izabrati suboptimalni plan upita ako su statistike zastarele kada se upit kompajlira. Statistike se obično ažuriraju ubrzo nakon toga. Upiti koji se kompajliraju nakon što se ažuriranja statistika završe će imati koristi od korišćenja ažuriranih statistika.

Korišćenje sinhronih statistika treba razmotriti kada vršimo operacije koje menjaju raspodelu podataka, kao što je brisanje tabele ili izvođenje grupnog ažuriranja velikog procenta redova. Ako ne ažuriramo statistike ručno nakon završetka operacije, korišćenje sinhronih statistika će osigurati da su statistike ažurirane pre nego što se izvrše upiti nad promenjenim podacima.

S druge strane, treba razmotriti korišćenje asinhronih statistika kako bismo postigli predvidive vremene odziva upita za sledeće scenarije:

* Ako naša aplikacija često izvršava isti upit, slične upite ili slične keširane planove upita. Vremena odziva upita mogu biti predvidljivija sa asinhronim ažuriranjem statistika nego sa sinhronim ažuriranjem statistika, jer optimizator upita može izvršavati dolazne upite bez čekanja ažuriranih statistika. To izbegava odlaganje nekih upita.
* Ako je naša aplikacija doživela vremensko ograničenje čekanja zahteva klijenta koji su uzrokovani time što jedan ili više upita čeka na ažurirane statistike. U nekim slučajevima, čekanje na sinhrone statistikame može dovesti do neuspeha aplikacija sa agresivnim tajm-autima.

Asinhrono ažuriranje statistika se vrši putem pozadinskog zahteva. Kada je zahtev spreman da upiše ažurirane statistike u bazu podataka, pokušava da dobije bravu (*lock*) modifikacije šeme (*schema modification lock*) na objektu metapodataka statistike. Ako druga sesija već drži bravu (*lock*) na istom objektu, asinhrono ažuriranje statistika se blokira dok se brava modifikacije šeme ne dobije. Slično tome, sesije koje treba da steknu bravu stabilnosti šeme (*Sch-S*) na objektu metapodataka statistika kako bi kompajlirale upit mogu biti blokirane od strane pozadinske sesije asinhronog ažuriranja statistika, koja već drži ili čeka da stekne bravu modifikacije šeme. Zbog toga, za radne zadatke sa veoma čestim kompajliranjem upita i čestim ažuriranjem statistika, korišćenje asinhronih statistika može povećati verovatnoću problema sa konkurencijom usled blokiranja brava.

Međutim, od verzije 16.x SQL Servera 2022, mogući problemi sa konkurencijom pri korišćenju asinhronog ažuriranja statistika mogu biti izbegnuti uz pomoć opcije *ASYNC\_STATS\_UPDATE\_WAIT\_AT\_LOW\_PRIORITY* koja se postavlja na nivou baze podataka. Kada je ova opcija omogućena, pozadinski zahtev će čekati da stekne bravu modifikacije šeme (*Sch-M*) i sačuva ažuriranu statistiku u posebnom redu niskog prioriteta, omogućavajući drugim zahtevima da nastave kompajliranje upita sa postojećim statistikama. Kada nijedna druga sesija ne drži zaključavanje (bravu) na objektu metapodataka statistike, pozadinski zahtev će dobiti bravu modifikacije šeme i ažurirati statistike. U retkom slučaju da pozadinski zahtev ne može da stekne bravu u okviru vremenskog ograničenja od nekoliko minuta, asinhrono ažuriranje statistika će biti prekinuto, i statistika se neće ažurirati dok se ne pokrene sledeće automatsko ažuriranje statistika ili dok statistike ne budu ažurirane ručno.

## **4.4. Opcija AUTO\_DROP**

U SQL Serveru, pre verzije 16 (SQL Servera 2022), ukoliko su statistike ručno kreirane od strane korisnika ili nekog *third-party* alata na korisničkoj bazi podataka, te statistike mogu blokirati ili ometati šematske promene koje korisnik možda želi.

Počevši od SQL Servera 2022 (verzija 16.x), opcija za automatsko brisanje (*AUTO\_DROP*) je podrazumevano omogućena na svim novim i migriranim bazama podataka. *AUTO\_DROP* svojstvo omogućava kreiranje statističkih objekata u modu u kojem kasnije promene šeme neće biti blokirane od strane statističkog objekta, već će statistike biti obrisane po potrebi. Na ovaj način, ručno kreirane statistike sa omogućenim automatskim brisanjem se ponašaju kao auto-kreirane statistike.

Pokušaji postavljanja ili uklanjanja svojstva automatskog brisanja na auto-kreiranim statistikama mogu izazvati greške. Auto-kreirane statistike uvek koriste automatsko brisanje. Neke rezervne kopije (*backup*-ovi) baza, kada se vraćaju (*restore*-uju), mogu imati ovo svojstvo netačno postavljeno sve dok sledeći put ne budu ažurirane statistike (ručno ili automatski). S toga, prilikom vraćanja baze podataka na SQL Server 2022 (verzija 16.x) sa prethodne verzije, preporučuje se izvršavanje procedure *sp\_updatestats* na bazi podataka kako bi se postavili odgovarajući metapodaci za funkciju automatskog brisanja statistika.

## **4.5. Opcija INCREMENTAL**

Kada je opcija *INCREMENTAL* pri kreiranju statistika postavljena na *ON*, kreirane statistike su statistike po particijama. Kada je postavljena na *OFF*, statističko stablo se briše i SQL Server ponovo računa statistike. Ova opcija je podrazumevano isključena.

Kada se novi segmenti (particije) dodaju velikoj tabeli, statistike bi trebalo ažurirati da bi obuhvatile nove segmente. Međutim, vreme potrebno za skeniranje cele tabele (opcija *FULLSCAN* ili *SAMPLE*) može potrajati dosta dugo. Takođe, skeniranje cele tabele nije neophodno jer su možda potrebne samo statistike za nove segmente. Opcija za inkrementalno ažuriranje kreira i skladišti statistike na nivou svake particije i kada se ažuriraju, osvežava samo statistike na onim particijama koje zahtevaju nove statistike.

# 5. Korisnički definisane statistike

## **5.1. Kreiranje statistike**

Ponekad statistike automatski generisane od strane SQL Servera neće biti optimalne za vrste upita koje izvodimo, u kom slučaju možemo kreirati svoje sopstvene statistike. Microsoft preporučuje da kreiramo statistike u sledećim okolnostima:

* Ako *Database Engine Tuning Advisor* to preporučuje.
* Ako upit vraća samo deo podataka iz ciljane kolone ili indeksa.
* Ako se uslov upita odnosi na više povezanih kolona u odvojenim indeksima.
* Ako optimizator upita nema potrebne statistike da bi generisao efikasan plan upita.

Na primer, pretpostavimo da želimo da postavimo upit za podatke u tabeli *AWSales* na osnovu vrednosti u koloni *LineTotal*, ali naši upiti uglavnom ciljaju samo redove sa vrednošću *LineTotal* većom od 100 dolara. Možemo kreirati filtrirani statistički objekat na osnovu tih vrednosti, kao što je prikazano u sledećem primeru:

IF EXISTS(SELECT name FROM sys.stats

WHERE name = N'TotalStats'

AND object\_id = OBJECT\_ID(N'dbo.AWSales'))

DROP STATISTICS dbo.AWSales.TotalStats;

GO

CREATE STATISTICS TotalStats ON dbo.AWSales(LineTotal)

WHERE LineTotal > 100

WITH FULLSCAN;

Primer 9. Upit za kreiranje statistike

Naredba *CREATE STATISTICS* omogućava kreiranje statističkog objekta na jednoj ili više kolona u tabeli ili indeksiranom pogledu. U ovom slučaju, ciljamo na kolonu *LineTotal* u tabeli *AWSales*, ali samo za vrednosti veće od 100, kako je naznačeno u *WHERE* klauzuli. Ovo predstavlja naš filter, pa se često koristi i termin "filtrirane statistike".

Naredba *CREATE STATISTICS* takođe uključuje klauzulu *WITH FULLSCAN*, koja specificira da bi svi redovi trebalo da budu pregledani prilikom kreiranja statistika. Alternativno, možemo specificirati da se određeni procenat podataka uzorkuje, umesto da se pregledaju svi redovi.

Ako bismo sada izvršili upit nad sistemskim pogledima *sys.stats* i *sys.stats\_columns*, koristeći istu *SELECT* naredbu kao pre, primetili bismo da je četvrti objekat statistika, *TotalStats*, dodat tabeli *AWSales*, kao što je prikazano u sledećoj tabeli koja predstavlja rezultate upita.

|  |  |  |  |
| --- | --- | --- | --- |
| StatsID | StatsName | StatsColID | ColumnName |
| 1 | PK\_\_AWSales\_\_C3905BAF9C6C0B09 | 1 | OrderID |
| 2 | \_WA\_Sys\_00000004\_39AD8A7F | 1 | ProductID |
| 3 | ixOrders | 1 | SalesOrderID |
| 3 | ixOrders | 2 | OrderDetailID |
| 4 | TotalStats | 1 | LineTotal |

Takođe, sada možemo izvršiti *DBCC SHOW\_STATISTICS* naredbu nad novim statističkim objektom da bismo videli kako je histogram mapiran:

DBCC SHOW\_STATISTICS (AWSales, TotalStats);

Sledeća tabela prikazuje rezultate izvršavanja naredbe. Primetite da linija 7 (označena na slici histograma), ima vrednost za *RANGE\_HI\_KEY* od 120 i vrednost za *EQ\_ROWS* od 99.

A screenshot of a computer

Description automatically generated

Slika 8. Rezultat izvršavanja *DBCC SHOW STATISTICS* naredbe za novi statistički objekat

Sada ćemo ponovo pokrenuti upit za tabelu *AWSales*, ovog puta uključujući *WHERE* klauzulu koja specificira da vrednost *LineTotal* treba da bude jednaka 120:

SELECT \*

FROM AWSales

WHERE LineTotal = 120

OPTION(RECOMPILE);

Primer 10. Upit za analizu histograma za graničnu vrednost novog statističkog objekta

Ovaj upit vraća 99 redova, što je ista vrednost u koloni *EQ\_ROWS* histograma i ista vrednost kao vrednost procenjenog broja redova (*Estimated Number of Rows*)u planu izvršenja, kao što je prikazano na sledećoj slici.

A screenshot of a computer screen

Description automatically generated

Slika 9. Procenjeni plan izvršenja za novi statistički objekat

Naravno, možemo eksperimentisati sa svojim upitima i upoređivati procene plana izvršenja sa histogramom. Zapravo, ovo je dobar način za sticanje boljeg osećaja za to kako se planovi izvršenja odnose na statistike. Samo, treba imati na umu da neki od proračuna optimizatora mogu biti malo složeniji, sa manje očiglednom logikom kako je došao od tačke A do tačke B.

## **5.2. Ažuriranje statistike**

Kao i prilikom generisanja statistika, SQL Server obično dobro održava njihovu ažuriranost. Međutim, u nekim slučajevima, moraćemo da intervenišemo i ažuriramo statistike ručno. Microsoft preporučuje da to radimo u sledećim okolnostima:

* Ako se naši upiti izvode sporije nego što bismo očekivali.
* Ako ubacujemo podatke u rastuće ili opadajuće kolone koje predstavljaju ključeve, kao što su *IDENTITY* kolone.
* Ako vršimo određene operacije održavanja, kao što je brisanje svih podataka iz tabele ili izvođenje grupnog umetanja podataka.

Na primer, pretpostavimo da dodajemo još jedan blok podataka u tabelu *AWSales* koristeći sledeću *INSERT* naredbu:

INSERT INTO dbo.AWSales

(SalesOrderID, OrderDetailID, ProductID, LineTotal)

SELECT SalesOrderID, SalesOrderDetailID, ProductID, LineTotal

FROM AdventureWorks2019.Sales.SalesOrderDetail

WHERE SalesOrderDetailID >= 60000;

Primer 11. Upit za unošenje dodatnih podataka u AWSales tabelu

Ako bismo ponovo izvršili sledeću *DBCC SHOW\_STATISTICS* naredbu, dobili bismo iste rezultate kao kada smo je izvršili u prethodnom odeljku:

DBCC SHOW\_STATISTICS (AWSales, TotalStats);

Na mom sistemu, *Updated* kolona u rezultatima zaglavlja prikazuje vrednost za datum 20. avgust 2023. godine i vreme kao 20:53, što predstavlja vreme pre nego što sam izvršio prethodnu *INSERT* naredbu, što ukazuje na to da histogram nije promenjen, uprkos dodavanju preko 60.000 novih redova.

Da bismo osigurali da naši upiti imaju najnovije statistike, možemo izvršiti *UPDATE STATISTICS* naredbu nad tabelom *AWSales*:

UPDATE STATISTICS dbo.AWSales

WITH FULLSCAN;

Primer 12. Upit za ažuriranje statistike

Takođe sam uključio i *WITH FULLSCAN* klauzulu da bih SQL Serveru rekao da skenira celu kolonu u svakom statističkom objektu prilikom ažuriranja statistika. Prema podrazumevanoj postavci, SQL Server koristi samo uzorak podataka umesto da izračunava statistike na osnovu svih podataka. Puna analiza obično pruža tačnije statistike, ali može rezultirati operacijama koje zahtevaju više resursa.

Nakon ažuriranja statistika, možemo ponovo pokrenuti *DBCC SHOW\_STATISTICS* naredbu, koja daje rezultate prikazane na sledećoj slici.

A screenshot of a computer

Description automatically generated

Slika 10. Rezultat izvršavanja *DBCC SHOW STATISTICS* naredbe za ažurirani statistički objekat

Možemo primetiti da je vreme ažurirano i da su koraci histograma definisali nešto drugačije opsege. Za malen skup podataka, ove razlike možda neće delovati značajno, ali kada počnemo da radimo sa milionima redova, ažuriranje statistika nakon određenih operacija može biti ključno za održavanje performansi našeg upita.

## **5.3. Brisanje statistike**

Naravno, korisnički definisane statistike se mogu i obrisati. Da bismo obrisali statistički objekat u SQL Serveru, možemo koristiti naredbu *DROP STATISTICS*. Ova naredba uklanja statistički objekat povezan sa navedenom kolonom ili skupom kolona u tabeli ili indeksiranom pogledu. Osnovna T-SQL sintaksa je sledeća:

DROP STATISTICS <ImeTabele>.<ImeStatistike>

## **5.4. Opcije za pregled statistika kod SQL Servera**

Sve ove sistemske i korisnički definisane statistike kod SQL Servera za određenu tabelu možemo pregledati kako pomoću SQL Server Management Studio-a, tako i pomoću T-SQL upita.

### 5.4.1. Korišćenje SSMS-a za predleg statistika

Za ovaj pristup se podrazumeva da smo povezani na instancu SQL Servera kroz SSMS alat, nakon čega možemo odabrati konkretnu bazu i tabelu (u mom slučaju *AdventureWorks2019.AWSales*) i videti sve dostupne statističke objekte u okviru *Statistics* direktorijuma.

A screenshot of a computer

Description automatically generated

Slika 11. Pregled statističkih objekata tabele kroz SSMS

Odavde možemo dobiti detalje o bilo kojoj određenoj statistici iz liste, odabirom opcije za pregled svojstava (desnim klikom na statistiku biramo *Properties*). To otvara svojstva statistike i prikazuje statističke kolone i datum poslednjeg ažuriranja za određenu statistiku.

A screenshot of a computer

Description automatically generated

Slika 12. Pregled svojstava određenog statističkog objekta

Zatim, klikom na *Details* opciju iz menija možemo dobiti prikaz distribucije vrednosti i učestalost svakog pojavljivanja različite vrednosti (histogram) za navedeni objekat.

A screenshot of a computer

Description automatically generated

Slika 13. Pregled detalja određenog statističkog objekta

### 5.4.2. Pregled statistika korišćenjem T-SQL upita

Za ovaj pristup možemo koristiti dinamički upravljački pogled (*DMV*) *sys.dm\_db\_stats\_properties* kako bismo videli osobine statistika za određeni objekat u trenutnoj bazi podataka. Uz pomoć sledećeg upita proverili smo statistike za tabelu *AdventureWorks2019.AWSales*.

SELECT sp.stats\_id,

name,

filter\_definition,

last\_updated,

rows,

rows\_sampled,

steps,

unfiltered\_rows,

modification\_counter

FROM sys.stats AS stat

CROSS APPLY sys.dm\_db\_stats\_properties(stat.object\_id, stat.stats\_id) AS sp

WHERE stat.object\_id = OBJECT\_ID('dbo.AWSales')

Primer 13. Upit za prikaz liste statističkih objekata

Dati upit vraća očekivane rezultate:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| stats\_id | name | filter\_definition | last\_updated | rows | rows\_sampled | steps | unfiltered\_rows | modification\_counter |
| 1 | PK\_\_AWSales\_\_C3905BAF9C6C0B09 | NULL | 2023-08-20 21:47:16.6500000 | 121317 | 121317 | 3 | 121317 | 0 |
| 2 | \_WA\_Sys\_00000004\_39AD8A7F | NULL | 2023-08-20 21:47:16.6633333 | 121317 | 121317 | 200 | 121317 | 0 |
| 3 | ixOrders | NULL | 2023-08-20 21:47:16.7100000 | 121317 | 121317 | 146 | 121317 | 0 |
| 4 | TotalStats | ([LineTotal]>(100)) | 2023-08-20 21:47:16.7300000 | 63605 | 63605 | 200 | 121317 | 0 |
| 5 | \_WA\_Sys\_00000005\_39AD8A7F | NULL | 2023-08-20 21:47:16.7633333 | 121317 | 121317 | 200 | 121317 | 0 |

A evo i šta kolone iz rezultata predstavljaju:

* stats\_ID: Jedinstveni ID statističkog objekta.
* name: Ime statistike.
* last\_updated: Datum i vreme poslednjeg ažuriranja statistike.
* rows: Prikazuje ukupan broj redova u vreme poslednjeg ažuriranja statistike.
* rows\_sampled: Daje ukupan broj uzoraka redova za statistiku.
* unfiltered\_rows: Ova kolona predstavlja ukupan broj redova u tabeli za koje se prikupljaju statistike, bez uzimanja u obzir bilo kog uslova filtera ili predikata.
* modification\_counter: Predstavlja vrednost koja prati broj promena napravljenih na statističkom objektu od poslednjeg ažuriranja statistike.

# 6. Literatura

1. *TutorialsPoint. "MS SQL Server - Overview". [*[*Online*](https://www.tutorialspoint.com/ms_sql_server/ms_sql_server_quick_guide.htm)*], [Accessed: 12-Aug-2023]*
2. *Microsoft Learn. "Editions and supported features of SQL Server 2022". [*[*Online*](https://learn.microsoft.com/en-us/sql/sql-server/editions-and-components-of-sql-server-2022?view=sql-server-ver16)*], [Accessed: 12-Aug-2023]*
3. *Microsoft Learn. "Introduction to SQL Server 2022". [*[*Online*](https://learn.microsoft.com/en-us/training/modules/introduction-to-sql-server-2022/)*], [Accessed: 12-Aug-2023]*
4. *Microsoft Learn. "Query processing architecture guide". [*[*Online*](https://learn.microsoft.com/en-us/sql/relational-databases/query-processing-architecture-guide?view=sql-server-ver16)*], [Accessed: 12-Aug-2023]*
5. *Sheldon, Robert (2016). "SQL Server Statistics Basics". Redgate Hub, [*[*Online*](https://www.red-gate.com/simple-talk/databases/sql-server/performance-sql-server/sql-server-statistics-basics/)*], [Accessed: 15-Aug-2023]*
6. *Microsoft Learn (2023). "Statistics". [*[*Online*](https://learn.microsoft.com/en-us/sql/relational-databases/statistics/statistics?view=sql-server-ver16)*], [Accessed: 16-Aug-2023]*
7. *Eset, Erkec (2021). "Fundamentals of SQL Server Statistics". [*[*Online*](https://www.sqlshack.com/fundamentals-of-sql-server-statistics/)*], [Accessed: 16-Aug-2023]*
8. *Gupta, Rajendra (2019). "SQL Server Statistics and how to perform Update Statistics in SQL". [*[*Online*](https://www.sqlshack.com/sql-server-statistics-and-how-to-perform-update-statistics-in-sql/)*], [Accessed: 17-Aug-2023]*

# 7. Listing

Slika 1. Optimizator upita 4

Slika 2. Karakteristične vrednosti u histogramu 9

Slika 3. Rezultat izvršavanja *DBCC SHOW STATISTICS* naredbe 11

Slika 4. Procenjeni plan izvršenja za vrednost koja jeste svoj red 12

Slika 5. Procenjeni plan izvršenja za vrednost koja nije svoj red 13

Slika 6. Procenjeni plan izvršenja za opseg vrednosti 14

Slika 7. Podešavanje statističkih opcija kroz dijalog svojstava baze podataka 16

Slika 8. Rezultat izvršavanja *DBCC SHOW STATISTICS* naredbe za novi statistički objekat 20

Slika 9. Procenjeni plan izvršenja za novi statistički objekat 21

Slika 10. Rezultat izvršavanja *DBCC SHOW STATISTICS* naredbe za ažurirani statistički objekat 22

Slika 11. Pregled statističkih objekata tabele kroz SSMS 23

Slika 12. Pregled svojstava određenog statističkog objekta 24

Slika 13. Pregled detalja određenog statističkog objekta 25

Primer 1. Kreiranje AWSales tabele 5

Primer 2. Dobijanje informacija o statističkim objektima pomoću sistemskih pogleda 6

Primer 3. Uključivanje uslova u upit pomoću *WHERE* klauzule 7

Primer 4. Kreiranje indeksa u *AWSales* tabeli 8

Primer 5. Korišćenje *DBCC SHOW STATISTICS* naredbe 11

Primer 6. Upit za analizu histograma za graničnu vrednost 12

Primer 7. Upit za analizu histograma za ne-graničnu vrednost 13

Primer 8. Upit za analizu histograma za opseg vrednosti 14

Primer 9. Upit za kreiranje statistike 19

Primer 10. Upit za analizu histograma za graničnu vrednost novog statističkog objekta 20

Primer 11. Upit za unošenje dodatnih podataka u AWSales tabelu 22

Primer 12. Upit za ažuriranje statistike 22

Primer 13. Upit za prikaz liste statističkih objekata 25