Originalni naučni rad UDK 004.655.3
Rad prihvaćen: 20. 1. 2017.

Preporuke za optimizaciju baza podataka

doc. dr Brankica Pažun [1]

[1] Fakultet za inženjerski menadžment, Univerzitet Union – Nikola Tesla,[2]

Apstrakt: Problem istraživanja rada jesu slabe performanse baze podataka i aplikacija, a koje nastaju usled neadekvatno napisanih SQL naredaba. U radu su predstavljene tehnike optimizacije sistema za upravljanje bazom podataka koje omogućavaju izvesna poboljšanja u odnosu na neoptimizovanu formu. U izabranom alatu se kao kriterijum poboljšanja koriste troškovi i vreme izvršavanja upita. Rezultati istraživanja su dati u vidu odgovarajućih preporuka optimizacije, čijom primenom se smanjuju troškovi i vreme izvršavanja upita, a što za posledicu ima slabije opterećenje diska, memorije i procesora, samim tim i efikasniji rad aplikativnog sistema.

Ključne reči: optimizacija, sistem za upravljanje bazom podataka, SQL

Some Recommendations in DBMS Optimization Process

Abstract: The problem of research is poor database and applications performance, which are caused by inadequate written SQL commands. This paper consists of DBMS optimization techniques that enable certain improvements compared to non-optimized version. The criterion used in research is cost and time of query execution. Research results are presented in the form of appropriate optimization recommendations, which implementation can reduce cost and time of query execution, therefore resulting in lower disk usage, as well as memory and processor usage, and thus the more efficient operation of an application system.

Key words: database management system, optimization, SQL

1. Uvod

Danas informacija predstavlja najznačajniji resurs kojim se raspolaže. Obim podataka se povećava svake godine, kao i količina podataka koja se skladišti po osobi. Pritom je neophodno obezbediti mehanizme kojima se na optimalan način ekstrahuju podaci od značaja. Primenom odgovarajućih tehnika optimizacije može se uticati na poboljšanje performansi naredaba koje se duže izvršavaju i nepotrebno zauzimaju resurse sistema, odnosno baze podataka. Operacije optimizatora podrazumevaju transformaciju SQL naredbe, procenu ukupnih troškova (pristupnih putanja i operacija spajanja), a koji zavise od procenjene potrošnje ulazno-izlaznih, procesorskih i memorijskih resursa, kao i generisanje plana izvršenja naredbe (Oracle, n.d.). Cilj rada je da se dođe do praktičnih preporuka u vezi sa optimizacijom, čiji efekti mogu na različite načine uticati na rad kompletnog sistema za upravljanje bazom podataka. Adekvatnim korišćenjem preporuka pri definisanju određenih naredaba optimizator kreira efikasniji plan izvršenja.

2. Tehnike optimizacije

Usled problema sa prostorom ili nedostatkom memorije bitan je segment upravljanja memorijom i mehanizam kompresije podataka, koji značajno mogu da optimizuju upotrebu hardvera. Ukoliko se neadekvatno rasporedi memorija, može doći do problema sa performansama, te u tom slučaju i tehnike optimizacije mogu naići na ograničenja pri pokušaju poboljšanja efikasnosti izvršenja naredaba. Sam potencijal hardvera može da iskoristi paralelizam koji se upotrebljava s ciljem ubrzanja izvršavanja naredaba podelom rada između više procesa.

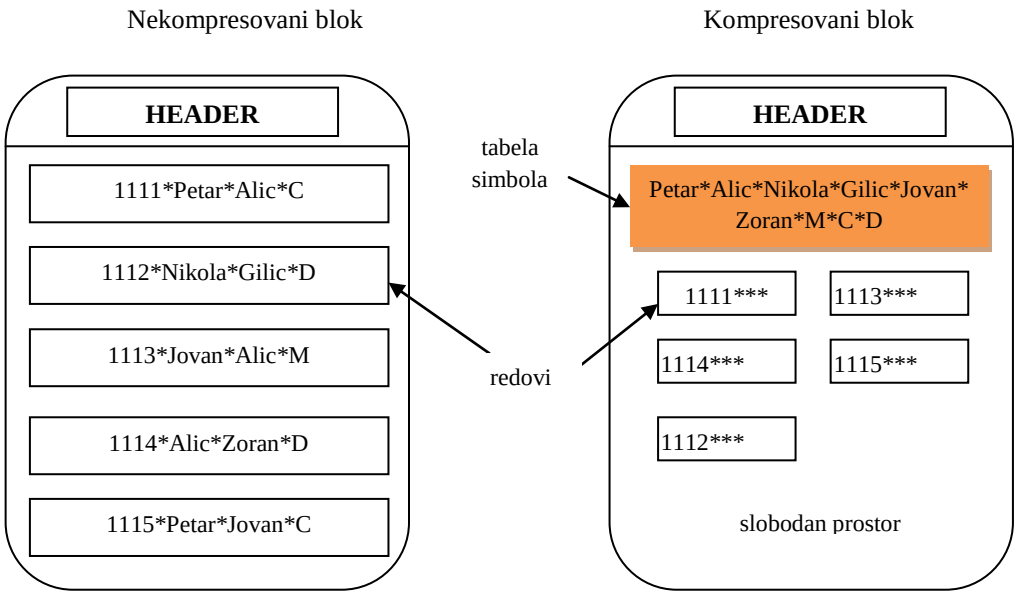
Kada se tabela sastoji iz velikog broja redova, neefikasna je i skupa operacija potpunog skeniranja radi nalaženja onih zapisa koji zadovoljavaju dati uslov. Tom prilikom se upotrebljava tehnika indeksiranja, koja, sa druge

[2] brankica.pazun@fim.rs

strane, može predstavljati najčešći uzrok loših performansi ukoliko nije efikasno iskorišćena prilikom definisanja upita. Da li je isplativije koristiti opciju "full table scan" ili indeks, zavisi od broja zapisa koji se vraćaju, kao i od samog hardvera i opterećenja servera. U opštem slučaju je isplativo koristiti bitmap indeks u slučaju da naredba vraća između 1% i 10% redova tabele. U tom slučaju se za svaku kolonu nad kojom je kreiran indeks, koristi bitmapa, odnosno niz bitova. Svaki bit u bitmapi odgovara jednoj vrednosti primarnog ključa. Funkcija konvertuje poziciju bita u odgovarajući identifikator zapisa, tako da bitmap indeksi omogućavaju istu funkcionalnost kao i b-stablo, samo što koriste drugačiji interni mehanizam. Bitmapa se sastoji od onoliko polja koliko ima različitih vrednosti u okviru indeksa. Najveće prednosti pri korišćenju indeksa se postižu kada nad jednom relacijom postoji više kreiranih bitmap indeksa, a istovremeno svi atributi imaju malu kardinalnost. Kreiranje više indeksa se primenjuje kod izvršavanja složenih naredbi. Ukoliko se u WHERE uslovu navedu sve kolone nad kojima je kreiran ovaj indeks, spajaju se pojedinačne bitmape i dobija se jedna koja predstavlja presek svih (Microsoft, n.d.).

Kompresija predstavlja tehniku upravljanja tabelama. Sam mehanizam koristi jedinstven algoritam čiji rad se zasniva na eliminaciji dvostrukih podataka iz blokova podataka. Nakon kompresije bloka, baza prvo računa broj ponavljanja vrednosti podataka u redovima. Kada prepozna podatke koji se ponavljaju, eliminiše ih iz zapisa i smesti blizu zaglavlja bloka. Za svaku ponavljajuću vrednost, u zaglavlju se zajedno sa podatkom upisuje i simbol kojim će biti predstavljen isti. Na slici 1. se može primetiti da su podaci iz zapisa izdvojeni i smešteni u poseban deo ispod sekcije "Header". Nakon što je blok kompresovan, dvostruke vrednosti se eleminišu dodavanjem jedne kopije u tabelu simbola. Svaki duplikat se potom menja odgovarajućim znakom iz tabele simbola. Kompresovani podaci se smeštaju u blokove podataka kao metapodaci, koji se ujedno koriste za prevođenje kompresovanih podataka u originalne podatke, a koji su se prvobitno nalazili u blokovima podataka. Pošto simboli zauzimaju manji prostor u odnosu na trenutne vrednosti simbola, sami zapisi zauzimaće manje memorije u poređenju sa prvobitnim redovima. Na slici 1. su prikazani nekompresovani i kompresovani blokovi kod Oracle baza, zajedno sa podacima u poljima.

Slika 1: Kompresija - Oracle SUBP



Izvor: prilagođeno prema Oracle (n.d.)

Prednosti tehnike kompresije kod Oracle sistema se ogledaju u sledećem:

- dovode do trostruko veće uštede prostora na disku;
- korišćenjem SSD diskova brzina pristupa može biti 300 puta veća u odnosu na pristup običnim diskovima;
- brze "full scan/ range scan" operacije - ukoliko optimizator odluči da koristi ove pristupne putanje pri kreiranju izvršnog plana, samo vreme izvršavanja je značajno kraće;
- smanjen mrežni saobraćaj - pošto su blokovi podataka kompresovani, eksterni paketi van mreže biće manji.

Star transformacija i particionisanje koje se u većini slučajeva poslednje primenjuje, predstavljaju česte tehnike u data warehouse sistemima, kao i upotreba bind varijabli koje kroz eliminisanje vremena potrebnog za parsiranje naredaba, poboljšavaju performanse. U Oracle sistemima postupak star transformacije, odnosno transformacije SQL naredbe, odvija se u dve faze: prvo se vraćaju redovi iz fact tabele korišćenjem bitmap indeksa kreiranih nad spoljnim ključevima iste, a potom se skup identifikatora zapisa spaja sa dimenzionom tabelom (Rittman, 2008). Sa druge strane, upotreba mehanizma hint omogućava uticaj na odluku optimizatora prilikom generisanja plana izvršenja tako što usmerava optimizator da izabere plan izvršenja sa određenim pristupnim putanjama, tj. operacijama.

3. Primena tehnika optimizacije

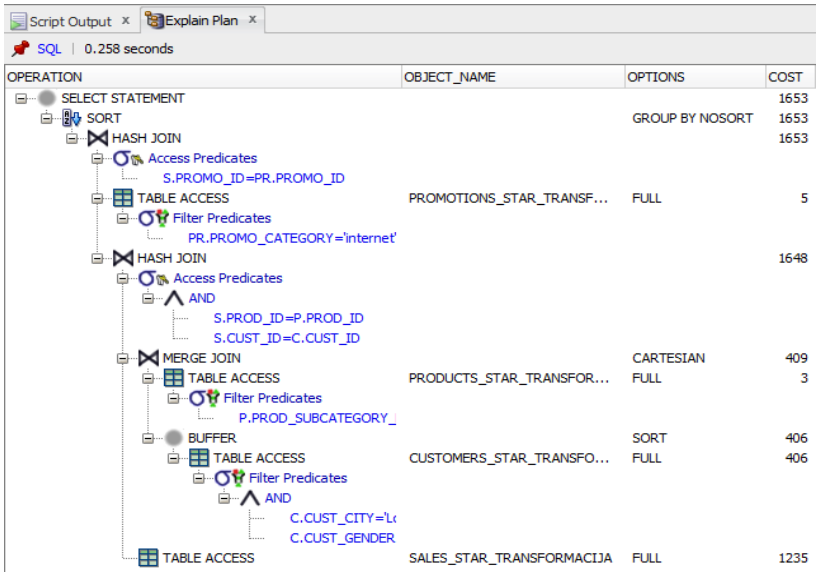
Upotrebom Oracle SQL developer alata primeniće se star transformacija koja se koristi nad naredbama koje se izvršavaju nad star šemom. Star šema se sastoji iz fact tabele i dve ili više dimenzionih tabela. Kreiraće se tabela sales_star_transformacija koja predstavlja fact tabelu i 3 dimenzione tabele, a što je uslov za postojanje star šeme. Fact tabela je preko spoljnih ključeva spojena sa dimenzionim tabelama.

```
create table sales_star_transformacija  as select * from sh.sales;
create table products_star_transformacija  as select * from sh.products;
create table customers_star_transformacija  as select * from sh.customers;
create table promotions_star_transformacija  as select * from sh.promotions;
alter table customers_star_transformacija add constraint customer_pk primary key (cust_id);
alter table products_star_transformacija add constraint prod_star_pk primary key (prod_id);
alter table promotions_star_transformacija add constraint promo_star_pk primary key (promo_id);
alter table sales_star_transformacija add constraint sales_prod_star_fk foreign key (prod_id)
    references products_star_transformacija;
alter table sales_star_transformacija add constraint sales_cust_star_fk foreign key (cust_id) references
    customers_star_transformacija;
alter table sales_star_transformacija add constraint sales_promo_star_fk foreign key (promo_id)
        references promotions_star_transformacija;
```

Pri izvršenju date naredbe trošak izvršavanja je 1653 (slika 2), usled čitanja kompletne tabele, tj. operacija FULL TABLE SCAN zbog nedostatka indeksa.

```
select sum (quantity_sold), pr.promo_category, c.cust_gender, c.cust_city,
p.prod_subcategory_desc
from sales_star_transformacija s, customers_star_transformacija c,
promotions_star_transformacija pr, products_star_transformacija p
where s.prod_id = p.prod_id
and s.cust_id = c.cust_id
and s.promo_id=pr.promo_id
and p.prod_subcategory_desc = 'Cameras'
and c.cust_city='Los Angeles'
and c.cust_gender='F'
and pr.promo_category='internet'
group by pr.promo_category, c.cust_gender, c.cust_city, p.prod_subcategory_desc;
```

Slika 2: Trošak izvršavanja naredbe 1653

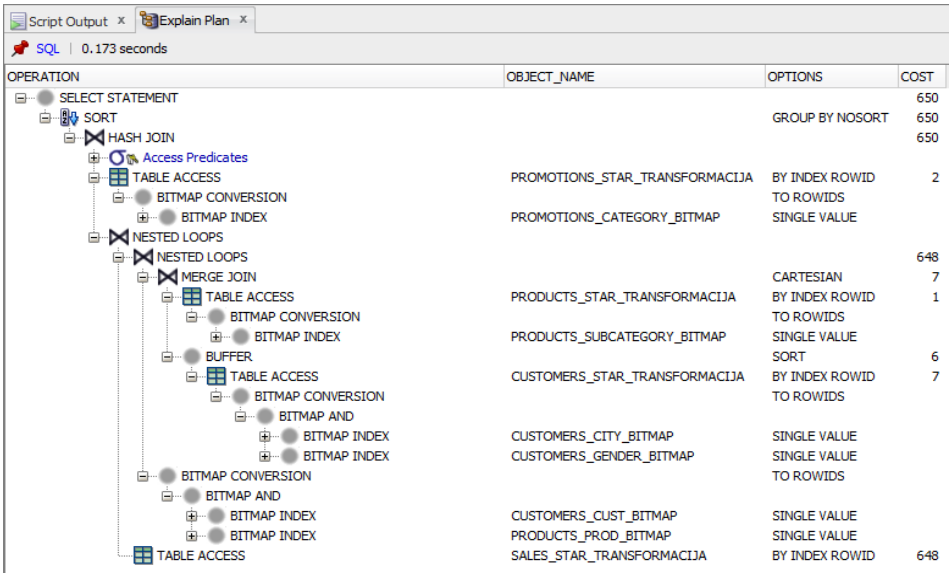


Izvor: autor

Posle kreiranja bitmap indeksa nad spoljnim ključevima fact tabele (sales_star_transformacija), trošak naredbe se smanjuje na 650 (slika 3). Zahvaljujući indeksima pri izvršenju naredbe vraćena je lista redova koji zadovoljavaju uslov (u vidu bitmape). Zatim je izvršen join fact tabele sa dimenzionim kako bi se vratile sve potrebne vrednosti iz dimenzionih tabela.

```
create bitmap index customers_cust_bitmap on sales_star_transformacija (cust_id);
create bitmap index promotions_promo_bitmap on sales_star_transformacija (promo_id);
create bitmap index products_prod_bitmap on sales_star_transformacija (prod_id);
create bitmap index customers_gender_bitmap on customers_star_transformacija (cust_gender);
create bitmap index customers_city_bitmap on customers_star_transformacija (cust_city);
create bitmap index products_subcategory_bitmap on products_star_transformacija
(prod_subcategory_desc);
create bitmap index promotions_category_bitmap on promotions_star_transformacija (promo_category);
```

Slika 3: Trošak izvršavanja naredbe 650

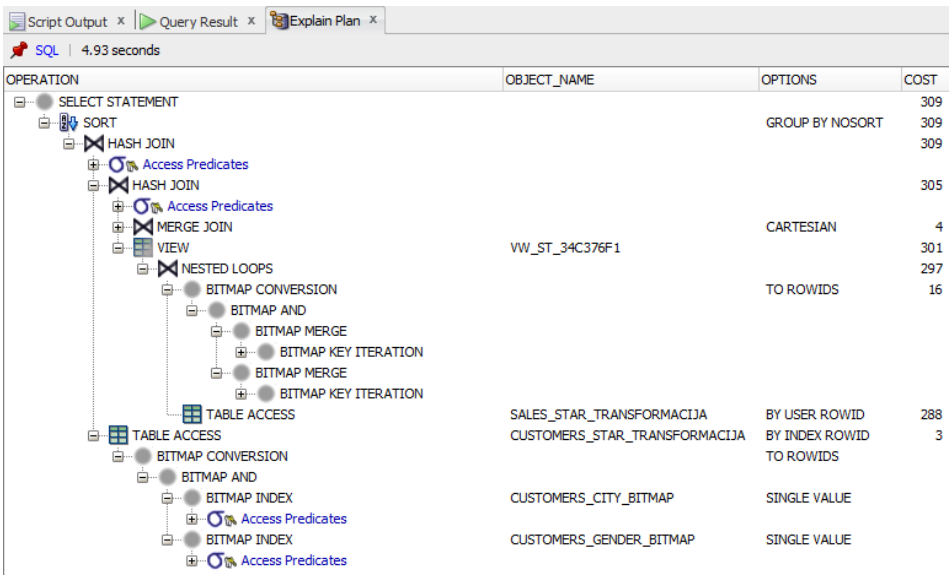


Izvor: autor

Posle omogućavanja star transformacije trošak se smanjuje na 309 (slika 4):

alter session set star_transformation_enabled='true';

Slika 4: Trošak izvršavanja naredbe 650

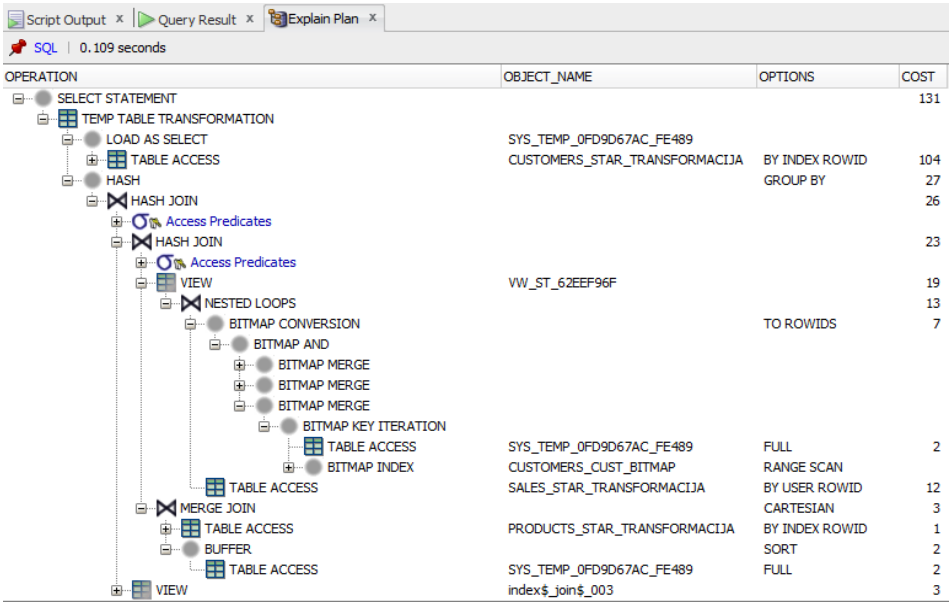


Izvor: autor

Još jednom dodatnom smanjenju na 131 doprinelo je sakupljanje statistike o datim kolonama, koje je pomoglo optimizatoru da generiše bolji plan izvršenja. Što su ažurnije statistike, vrednost je tačnija, odnosno, odluke optimizatora su bolje.

```
analyze table sales_star_transformacija compute statistics for table for all indexes for all indexed
        columns;
analyze table customers_star_transformacija compute statistics for table for all indexes for all
        indexed columns;
analyze table products_star_transformacija  compute statistics for table for all indexes for all
        indexed columns;
analyze table promotions_star_transformacija compute statistics for table for all indexes for all
        indexed columns;
```

Slika 5: Trošak izvršavanja naredbe 131



OPERATION	OBJECT_NAME	OPTIONS	COST
SELECT STATEMENT			131
TEMP TABLE TRANSFORMATION			
LOAD AS SELECT	SYS_TEMP_0FD9D67AC_FE489		
TABLE ACCESS	CUSTOMERS_STAR_TRANSFORMACIJA	BY INDEX ROWID	104
HASH		GROUP BY	27
HASH JOIN			26
Access Predicates			
HASH JOIN			23
Access Predicates			
VIEW	VW_ST_62EEF96F		19
NESTED LOOPS			13
BITMAP CONVERSION		TO ROWIDS	7
BITMAP AND			
BITMAP MERGE			
BITMAP MERGE			
BITMAP MERGE			
BITMAP KEY ITERATION			
TABLE ACCESS	SYS_TEMP_0FD9D67AC_FE489	FULL	2
BITMAP INDEX	CUSTOMERS_CUST_BITMAP	RANGE SCAN	
TABLE ACCESS	SALES_STAR_TRANSFORMACIJA	BY USER ROWID	12
MERGE JOIN		CARTESIAN	3
TABLE ACCESS	PRODUCTS_STAR_TRANSFORMACIJA	BY INDEX ROWID	1
BUFFER		SORT	2
TABLE ACCESS	SYS_TEMP_0FD9D67AC_FE489	FULL	2
VIEW	index$_join$_003		3

Izvor: autor

4. Preporuke za optimizaciju baze

Različite naredbe kreiraju drugačije planove izvršenja, a samim tim se razlikuju vreme i trošak izvršavanja, bez obzira na istovetan ishod. Potrebno je znati u kome trenutku i na koji način je neophodno optimizovati naredbu da bi se dobio dovoljno dobar, odnosno optimalan plan izvršenja. Sama primena tehnika optimizacija ne bi imala smisla bez adekvatnog upravljanja memorijom. Memorijske resurse bi itrebalo prebaciti na one komponente kojima je to neophodno.

U opštem slučaju najčešći uzrok loših performansi je neupotreba ili preterana upotreba indeksa. U slučaju obavljanja česte pretrage po određenim kolonama, poželjno je nad njima kreirati indekse. Ukoliko se u bazi definiše suviše mali broj indeksa, pronalaženje podataka će trajati duže nego što bi trebalo. Sa druge strane, ukoliko se definiše suviše veliki broj indeksa, naredbe nad podacima, kao što su unos i ažuriranje će trajati duže nego što bi trebalo.

Ukoliko postoji potreba za izvršavanjem mnogo komandi iste stukture, sa rezultirajućom promenom vrednosti samo jednog atributa (primera radi: Proizvod_id=5, Proizvod_id=7), najbolje je koristiti bind varijable. Drugi slučaj njihove primene jeste u radu sa rekurzivnim funkcijama koje u svojoj WHILE petlji uzimaju različite vrednosti iz tabele i pritom višestruko izvršavaju upit iste strukture. U ovakvim slučajevima je korišćenje bind varijabli više nego potrebno, jer se sa velikom verovatnoćom može desiti da vreme parsiranja svih sličnih upita bude duže od vremena izvršavanja upita (posebno ukoliko postoje paralelni zahtevi). U aplikacijama namenjenim masovnom skladištenju podataka (data warehouse) češće se upotrebljava keširanje naredbe kao i same tabele, umesto bind mehanizma.

Mehanizam paralelizma se predlaže u slučaju izvršavanja složenih naredaba, pri čemu je vreme izvršavanja duže (npr. duže od 10 sekundi). U suprotnom, trošak korišćenja paralelizma može biti dosta veći, a i sama primena tehnike uključuje kvalitetnije memorijske resurse i CPU.

Kompresija se preporučuje u slučaju nedostatka prostora za skladištenje podataka, čime se može ostvariti racio kompresije i do 3.5:1. Pristupa se većoj količini podataka na istom prostoru, više podataka može da se "pročita", samim tim se i izvršavanje naredaba ubrzava. Razlog tome se ogleda u manjem broju blokova/ strana koje je potrebno "pročitati". Pošto manji broj strana/blokova ostaje u keš memoriji, više prostora ostaje za druge blokove, što opet doprinosi poboljšanju performansi.

Kad je reč o tabelama sa ogromnim količinama podataka, od čega je većina podataka istorijska, predlaže se upotreba particionisanja. Istorijske podatke, odnosno, podatke koji se retko upotrebljavaju, treba skladištiti na veće particije i smestiti na sporije diskove. Noviji podaci se pohranjuju na brže diskove, manje particije, kako bi, zbog čestog pristupa tim podacima, samo izvršavanje naredbi bilo mnogo kraće.

Star transformacija je tehnika koja se preporučuje posebno u radu sa data warehouse sistemima, gde postoji potreba za izvršavanjem naredaba nad star šemom. Upotrebljava se kada se veliki broj dimenzionih tabela spaja sa fact tabelom, omogućavajući kraće vreme izvršavanja naredbi.

Na kraju, u slučaju neuspelog pokušaja primene neke od tehnika optimizacije, u smislu neadekvatnog načina upotrebe ili se smatra da dobijeni plan izvršenja nije optimalan, upotrebljava se mehanizam hint koji će uticati na generisanje drugačijeg izvršnog plana, jer utiče na optimizator da koristi drugu pristupnu putanju.

4. Zaključak

Krajnji korisnici danas posežu za trenutnim odgovorima. Iz razloga bržeg pristupa podacima, neophodno je razumeti upravljanje performansama. Od svih problema u vezi sa performansama sistema za upravljanje bazama podataka, 75% do 80% se mogu odnositi na neadekvatno i nekvalitetno napisane SQL naredbe. Komande koje se izvršavaju treba napisati na odgovarajući u cilju osiguranja optimalne performanse baze podataka i aplikacije. Svi sistemi za upravljanje bazama podataka sadrže optimizator koji pri izvršavanju upita generiše plan izvršenja sa ciljem obezbeđenja što boljih performansi. Međutim, optimizator često ne generiše ni približno dobar plan, zbog čega se naredbe duže izvršavaju, što je i predmet istraživanja ovog rada.

Opisani su neki od mehanizama optimizacije u Oracle sistemima. Zahvaljujući tehnikama došlo se do praktičnih preporuka čijom primenom optimizator kreira efikasnije planove izvršenja, a time i poboljšava performanse naredaba. Samo poboljšanje se ogleda u kraćem vremenu izvršenja naredbi, manjem opterećenju procesora, odnosno, kvalitetnijem upravljanju memorijskim resursima.

Kako pri izvršavanju naredaba ne bi dolazilo do ograničenja usled neoptimizovanog hardvera, predstavljeni su upravljanje memorijom i kompresija podataka. Opisani su indeksi koji se koriste s ciljem kraćeg vremena pristupa tabelama pri čestom pretraživanju podataka. Predstavljene su bind varijable, čijom upotrebom se poboljšavaju performanse korišćenjem istih planova izvršenja. Usled potrebe za jednostavnijim upravljanjem velikom količinom podataka, predstavljene su tehnike star transformacija i particionisanje tabela, koje se najviše koriste u data warehouse sistemima. U isto vreme je navedena i mogućnost upotrebe hinta koji može da usmeri optimizator na korišćenje određene tehnike.

U radu je predstavljen primer sa izvršnim planom i troškom izvršenja naredbe na početku, a zatim i stanje posle određenih izmena, tj. implementacije tehnika optimizacije. Prikazana je primena bitmap indeksa, sakupljanje statistike i upotreba star transformacije. Nakon svakog koraka predstavljen je plan izvršenja i praćen je trošak izvršenja naredbe, sa trendom opadanja.

Većina baza podataka podržava neke optimizacione tehnike radi bržeg pristupa podacima. Ideja o optimizaciji u svim sistemima za upravljanje bazama podataka je identična, sa razlikom u načinu implementacije.

Literatura

1. Fiorillo, C. (2012). *Oracle Database 11gR2 Performance Tuning Cookbook.* Packt publishing.
2. Oracle. (n.d.). Introduction to the optimizer. Preuzeto 22. decembra 2016, sa http://docs.oracle.com/cd/B10500_01/server.920/a96533/optimops.htm#51003

3. Microsoft. (n.d.). Kreiranje i upotreba indeksa radi poboljšanja performansi. Preuzeto 22. decembra 2016, sa http://office.microsoft.com/sr-latn-cs/access-help/kreiranje-i-upotreba-indeksa-radi-poboljsanja-performansi-HA010341594.aspx
4. Oracle. (n.d.). Compression. Preuzeto 29. decembra 2016, sa http://www.oracle.com/technetwork/articles/oem/11g-compression-198295.html
5. Rittman, M. (2008). So How Do Star Transformations Actually Work? Preuzeto 29. decembra 2016, sa http://www.rittmanmead.com/2008/12/so-how-to-star-transformations-actually-work/