Univerzitet u Nišu, Elaktronski fakultet

Katedra za računarstvo

**Obrada upita Oracle baze podataka**

Student : Saška Radenković 1132

Profesor : Prof. dr Aleksandar Stanimirović

SaDRŽAJ:

[Uvod 3](#_Toc69214721)

[Postupak obrade upita 4](#_Toc69214722)

[SQL Parsiranje 5](#_Toc69214723)

[Teško raščlanjivanje - Hard Parsing 5](#_Toc69214724)

[Optimizator upita 9](#_Toc69214725)

[Plan izvrsenja generisan Optimizatorom 10](#_Toc69214726)

[Komponente optimizatora 11](#_Toc69214727)

[Transformacija upita (Query Transformation) 12](#_Toc69214728)

[Procenjivač (Estimation) 12](#_Toc69214729)

[***Selektivnost*** 13](#_Toc69214730)

[***Kardinalnost*** 14](#_Toc69214731)

[***Trošak*** 15](#_Toc69214732)

[*SQL Row Source Generation* 16](#_Toc69214733)

[SQL Izvršenje upita 17](#_Toc69214734)

[Zaključak 21](#_Toc69214735)

[Reference 22](#_Toc69214736)

# Uvod

Glavni cilj stvaranja baze podataka je skladištenje povezanih podataka na jednom mestu, pristup i upravljanje podacima kada i kako to korisnik zahteva. Pristup podacima i manipulacija njima treba da se obavlja efikasno, tako da obezbeđuje lak i brz pristup. SQL je jezik visokog nivoa stvoren za izgradnju mosta između korisnika i DBMS-a za njihovu komunikaciju. Obrada upita i optimizacija su osnovni, ako ne i kritični delovi bilo kog DBMS-a. Da bi se efikasno koristili, rezultati upita moraju biti dostupni u vremenskom roku koji je potreban korisniku koji šalje podatke - bilo da je reč o osobi ili nekom drugom zasebnom i odvojenom DBMS-u. [1]

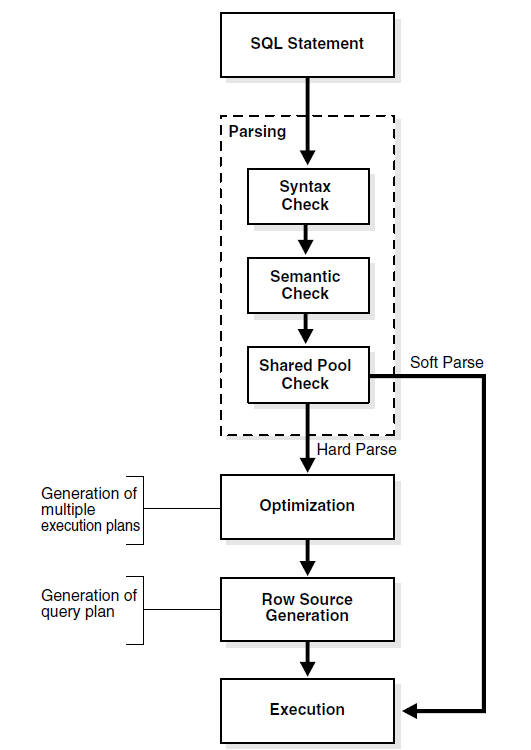
Osnovni sistemi u DBMS-u ne razumeju SQL, kao posledica toga mora postojati jezik na niskom nivou koji ovi sistemi mogu da razumeju. Upit napisan u SQL-u se pretvara u jezik niskog nivoa pomoću relacione algebre koju sistem može razumeti. DBMS zahteva od svojih korisnika da upišu upit u SQL-u, proverava kod koji je napisao korisnik, a zatim ga pretvara u jezike niskog nivoa. DBMS bira najbolju putanju izvršenja i izvršava upit. Procesi koje sprovodi baza podataka su zajedno poznati kao obrada upita. [1]

Obrada upita je predstavlja razbijanja jezika visokog nivoa na jezik niskog nivoa koji mašina može da razume i izvrši traženu radnju za korisnika. Ovaj zadatak izvršava Query processor u DBMS-u. U sklopu obrade upita, nakon prevodjenja na jezik niskog nivoa, proverava se tačnost teksta samog upita. Izdvajaju se tokeni i proverava se semantička i sintaksna ispravnost izjave. Nakon provere teksta same izjave, baza podataka određuje najefikasniji način izvršenja zadate izjava na osnovu određenih parametara i cene upita, pri čemu se optimalna izjava predaje na izvršenje.

U nastavku rada su predstavljene faze obrade upita, pri čemu se daje detaljan prikaz svake podfaze.

# Postupak obrade upita

Faze SQL obrade upita mogu se podeliti na par opštih faza: rasčlanjivanje (parsiranje), optimizaciju, proizvodnju redova i izvršenje.



Slika 1: Koraci u SQL obradi upita [2]

Slika 1 predstavlja dijagram toka izvršenja operacija prilikom obrade upita. Zavisno od toga da li se radi o ***Soft Parsing-u*** ili ***Hard Parsing-***u prolazi se kroz različite faze obrade. Dijagram prikazuje kako se upit obrađuje u bazi podataka da bi se prikazao rezultat. Kada se upit preda bazi, prima ga prevodilac upita. Zatim skenira upit i deli ga na pojedinačne tokene. Kada su tokeni generisani, parser proverava ispravnost. Upiti se transformišu u različite relacione izraze, relaciono stablo i relacione grafikone- ***plan upita***. Optimizator generiše različite ***planove izvršenja*** za dati plan upita. Plan izvršenja upita zatim predstavlja najbolje i optimizovano resenje plana izvršenja. Procesor naredbi koristi ovaj plan izvršenja za preuzimanje podataka iz baze podataka i vraća rezultat. [2]

SQL upit sačinjen je od različitih ulaza, odnosno može se sastojati iz tabela, funkcija ili izraza, ova karakteristika upita dovodi do mogućnosti da se jedan upit može izvršiti na više različitih načina. Za izvršenje upita u najkraćem roku neophodno je da se on izvrši na najoptimalniji način. Raščlanjivanje upita je postupak kojim se vrši donošenje odluke o najoptimalnijem izvršenju upita, na osnovu računice na koliko različitih načina se može izvršiti upit. Svaki upit mora biti raščlanjen najmanje jednom. Raščlanjivanje upita vrši se unutar baze podataka pomoću komponente ***Optimizer***. Optimizer procenjuje mnoge različite atribute datog upita (broj uključenih tabela, indekse, o kakvim izrazima je reč, itd.). Uzimajući u obzir sve ove ulaze, Optimizer odlučuje o najboljem mogućem načinu izvršenja upit, dok se ove informacije čuvaju u okviru SGA u kešu biblioteke.

## SQL Parsiranje

Postoje dva moguća stanja za obradu podataka upita. Jedna način obrade predstavlja postojanje upita unutar keš biblioteke, dok se u drugim slučajevima optimalno izvršenje upita ne može naći. Područje memorije unutar keša biblioteke u kojem se čuvaju informacije o obradi upita naziva se ***kursor***. Ako se kursor za višekratnu upotrebu pronađe u kešu biblioteke može se upotrebiti za izvršenje upita. Ovakav princip se naziva Lako raščlanjivanje (eng. ***Soft Parsing)***. Kada nije moguće pronaći kursor za ponovnu upotrebu ili ako upit nikada ranije nije izvršen, potrebno je izvršiti optimizaciju upita – Teško raščlanjivanje (eng. ***Hard Parsing).*** [3]

### Teško raščlanjivanje - Hard Parsing

Teško raščlanjivanje znači da ili kursor nije pronađen u kešu biblioteke ili je pronađen, ali je iz nekog razloga onesposobljen. Primena Hard Parsing-a bi značila da optimizator treba da radi kako bi se osigurao najoptimalniji plan izvršenja upita. Optimizator to čini promatrajući sve moguće unose koje mu daje korisnik (uključuje prisustvo/odsustvo bilo kojih indeksa, izraza ili funkcija primenjenih na kolone, bilo da je reč o upitu za pridruživanje ili ne, bilo kojim navedenim nagoveštajima itd.). Sve ulazne informacije su od velike važnosti i prisustvo ili odsustvo bilo kakvih takvih unosa može promeniti plan izvršenja.

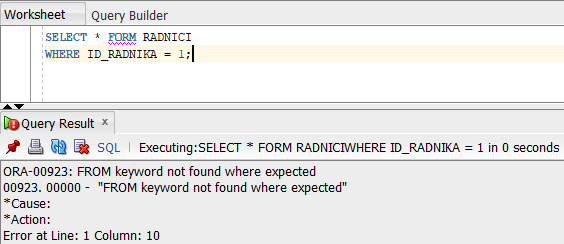
Pre nego što se započne postupak pronalaženja najboljeg plana za upit, postoje neki zadaci koji se prethodno moraju završiti. Ovi zadaci se izvršavaju više puta, čak i ako se isti upit izvršava N puta u istoj sesiji:

1. Provera sintakse
2. Provera semantike
3. Heširanje teksta upita i generisanje para ključ / vrednost heša

Pre nego što se uradi bilo šta vezano za upit, važno je da sintaksa upita bude tačna.

*Koja je svrha pokušaja pronalaženja najboljeg mogućeg načina za izvršavanje upita kada on uopšte ne može da se izvrši zbog ključne reči koja nedostaje?*

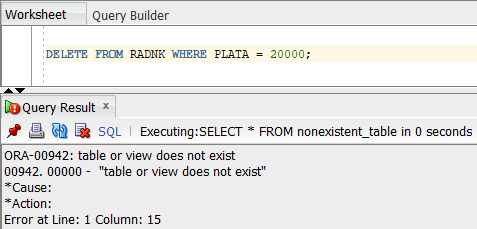
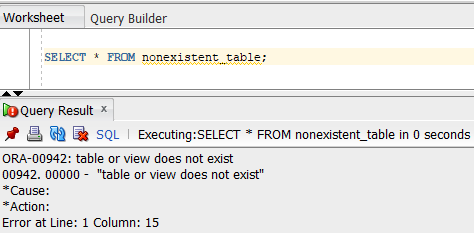
1. **Sintaksa** - Baza podataka proverava da li je upit napisan sa ispravnom sintaksom ili ne, kao i da li korisnik koji izvršava upit ima odgovarajuće dozvole za osnovne objekte. Ako bilo koja od ove dve provere ne uspe, postupak izvršenja upita se prekida. Oracle Database mora proveriti sintaksičku valjanost svake SQL naredbe. Izjava koja krši pravilo za dobro oblikovanu SQL sintaksu ne uspeva u proveri. Primer: sledeća izjava nije validna jer je ključna reč **FROM** pogrešno napisana kao **FORM**: [3]



Slika 2: Sintaksno neispravna izjava

Sintaksa se proverava upoređivanjem celokupnog teksta upita sa podržanim ključnim rečima za verziju baze podataka koja se trenutno koristi.

1. **Semantika** iskaza predstavlja njeno značenje. Dakle, semantička provera određuje da li je iskaz smislen, na primer, da li postoje objekti i kolone navedeni u iskazu. Sintaksički tačna izjava može biti semantički nevalidna. Primer prikazuje sintaksički validu izjavu dok je semantički pogrešna: [3]



Slike 3.a i 3.b prikazuju semantički nevalidnu izjavu

Semantika proverava da li se koristi tačno ime objekta, kao i da li korisnik ima odgovarajuće privilegije za izvršavanje upita na objektima koji se koriste u izrazu.

1. **Shared Pool Check** - Nakon prolaska oba koraka (sintaksna i semantička analiza), sledeći i najvažniji zadatak je da se potraži zadati upit u SGA, tačnije u kešu biblioteke. Budući da će to biti memorijska struktura, najbolji način pretraživanja u memoriji bio bi putem mehanizma heširanja. Oracle baza podataka uzima unos teksta upita i koristi ga za generisanje heš vrednosti za upit. Hash vrednost izraza predstavlja SQL ID. Kada korisnik pošalje SQL izraz, baza podataka pretražuje zajedničko SQL područje kako bi utvrdila da li postojeći raščlanjeni izrazi imaju istu heš vrednost.

Oracle Database automatski određuje da li je SQL blok koji se izdaje identičan drugom izrazu koji se trenutno nalazi u deljenom spremištu. Da bi uporedio tekst SQL izraza sa postojećim SQL izrazima u deljenom spremištu, Oracle Database izvodi sledeće korake:

* Tekst SQL izraza je heširan.

Ako ne postoji odgovarajuća hash vrednost, tada SQL izraz trenutno ne postoji u deljenom spremištu i vrši se tesko parsiranje.

* Ako postoji odgovarajuća heš vrednost za postojeći SQL izraz u deljenom spremištu, tada se tekst podudarne izjave upoređuje sa tekstom heširane naredbe da bi se proverilo da li su identični.

Tekst SQL izraza mora biti identičan, uključujući razmake, velika slova i komentare. Na primer, sledeće izjave ne mogu koristiti isto zajedničko SQL područje:

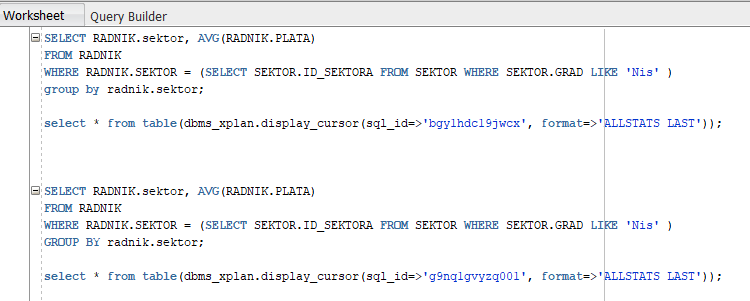
SELECT \* FROM employees;

SELECT \* FROM Employees;

SELECT \* FROM employees;

Slika 4: Primer nepodudaranja teksta izjave

Primer:



Slika 5: Izjave koje imaju istu heš vrednost dok je tekst izjave drugaciji

Slika 5 predstavlja primer u kome dve izjave imaju istu hash vrednost nakon heširanja, međutim SQL\_ID izjava je drugaciji zbog toga sto se ne tretiraju kao podudarne izjave zbog razlike u tekstu sql upita (razlika u velikim i malim slovima ključne reči GROUP BY)

Takođe, SQL izrazi koji se razlikuju samo u literalima ne mogu koristiti isto zajedničko SQL područje. Na primer, sledeće izjave se ne rešavaju u istom SQL području:

SELECT count(1) FROM employees WHERE manager\_id = 121;

SELECT count(1) FROM employees WHERE manager\_id = 247;

Slika 6: Razlika teksta u literalu

* Objekti na koje se poziva izdata izjava upoređuju se sa referenciranim objektima svih postojećih izraza u zajedničkom spremištu kako bi se osiguralo da su identični.

Referencije na objekte šeme u SQL izrazima moraju se odnositi na isti objekat u istoj shemi. Na primer, ako dva korisnika izdaju isti SQL izraz, ali svaki od njih ima svoju tabelu zaposlenih, tada se ova izjava ne smatra identičnom, jer se izjava odnosi na različite tabele za svakog korisnika.

Kada se data heš vrednost i tekst izjave tačno podudaraju sa sličnom izjavom u predmemoriji biblioteke, izjava se može izvršiti kao Soft Parsed izjava. Ali ako nije, upit mora biti teško raščlanjen, to zapravo znači da Oracle baza podataka treba da proceni najbolji mogući način izvršavanja upita. Ova optimizacija se izvodi u tri dela.

## Optimizator upita

Optimizator upita je ugrađeni softver za baze podataka koji određuje najefikasniji metod za pristup SQL izjavi traženim podacima.

Optimizator pokušava da generiše najoptimalniji plan izvršenja za SQL izraz, bira plan sa najnižom cenom među svim razmatranim planovima kandidata. Optimizator koristi raspoložive statistike za izračunavanje troškova. Kada se izračunavaju troškova u obzir se uzimaju faktori izvršavanja upita kao što su I/O, CPU i komunikacija.

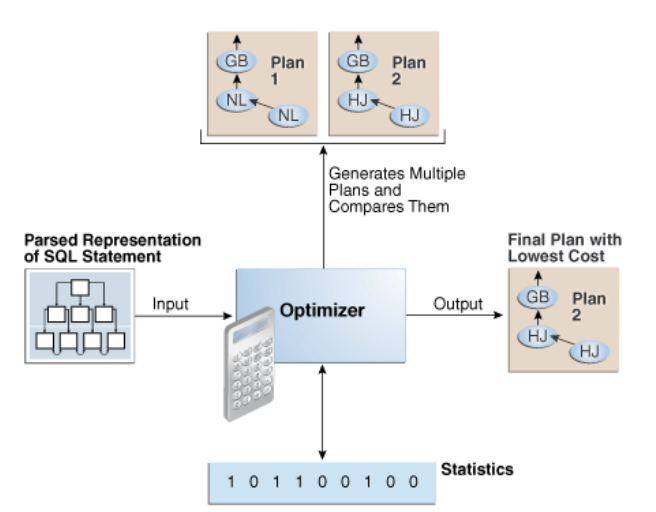
*Primer: Upit može tražiti informacije o zaposlenima koji su menadžeri. Ako statistika optimizatora ukazuje da je 80% zaposlenih menadžer, tada optimizator može odlučiti da je skeniranje kompletne tabele najefikasnije. Međutim, ako statistika ukazuje na to da je vrlo mali broj zaposlenih menadžera, čitanje indeksa praćenog pristupom tabele može biti efikasnije od potpunog skeniranja tabele.*

Baza podataka ima na raspolaganju mnogo internih statistika i alata, pri čemu je optimizator obično u boljoj poziciji od korisnika da odredi optimalan način izvršavanja izraza, ova činjenica predstavlja glavni razlog da svi SQL izrazi koriste optimizator. Baza podataka optimizuje svaki SQL izraz na osnovu statistike prikupljene o podacima kojima se pristupa. Optimizator određuje optimalan plan za SQL izraz ispitivanjem metoda višestrukog pristupa, kao što su skeniranje pune tabele ili skeniranje indeksa, različite metode spajanja kao što su ugneždene petlje i heš spajanja, različiti redosledi pridruživanja i moguće transformacije.

Za dati upit i okruženje, optimizator dodeljuje relativni numerički trošak svakom koraku mogućeg plana, a zatim te vrednosti faktorizuje da bi stvorio ukupnu procenu troškova za plan. Nakon izračunavanja troškova alternativnih planova, optimizator bira plan sa najnižom procenom troškova. Iz tog razloga, optimizator se ponekad naziva i optimizator zasnovan na troškovima (CBO). [5]

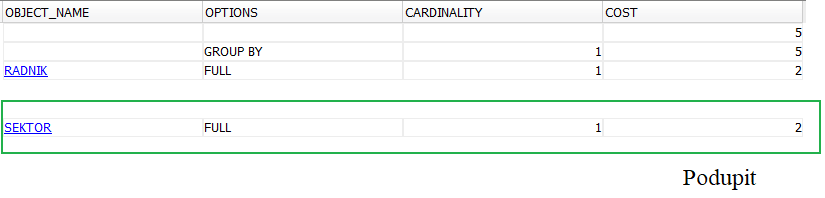
### Plan izvrsenja generisan Optimizatorom

Na sledećoj slici, optimizator generiše dva moguća plana izvršenja za ulazni SQL izraz, koristi statistiku za procenu njihovih troškova, upoređuje njihove troškove, a zatim bira plan sa najnižom cenom.

**

Slika 7: Izbor plana izvršenja [5]

Za svaki blok upita, optimizator generiše podplan upita. Baza podataka optimizuje blokove upita odvojeno odozdo prema gore. Dakle, baza podataka prvo optimizuje najugnježdeniji blok upita i generiše podplan za njega, a zatim generiše spoljni blok upita koji predstavlja ceo upit. Broj mogućih planova za blok upita proporcionalan je broju objekata u klauzuli FROM. Ovaj broj eksponencijalno raste sa brojem objekata.



*Primer, mogući planovi za spajanje pet stolova su znatno veći od mogućih planova za spajanje dve tabele.*

***Analogija za optimizator***: Jedna od analogija optimizatora je onlajn savetnik za putovanja.

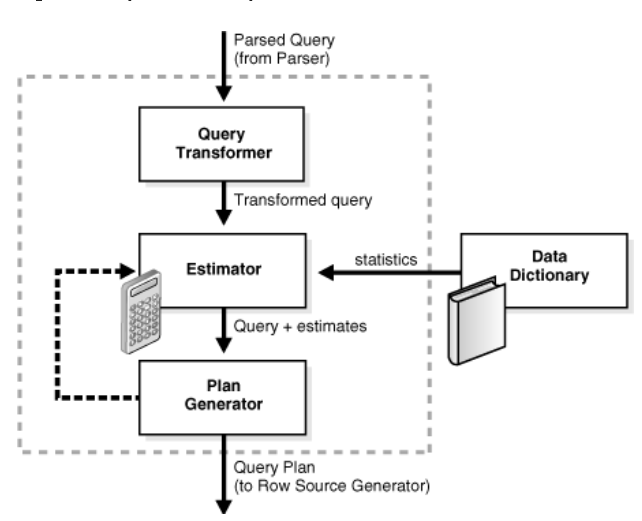
Biciklista želi da zna najefikasniju biciklističku rutu od tačke A do tačke B. Upit je poput direktive „Treba mi najefikasnija ruta od tačke A do tačke B“ ili „Treba mi najefikasnija ruta od tačke A do tačke B putem tačke C. " Savetnik za putovanje koristi interni algoritam koji se oslanja na faktore kao što su brzina i poteškoće, da bi odredio najefikasniju rutu. Biciklista može uticati na odluku savetnika za putovanje koristeći direktive poput „Želim da stignem što brže“ ili „Želim najlakšu moguću vožnju“.

U ovoj analogiji, plan izvršenja je moguća ruta koju generiše savetnik za putovanje. Interno, savetnik može podeliti celokupnu rutu u nekoliko podplanova i izračunati efikasnost za svaki podplan posebno. Na primer, savetnik za putovanje može da proceni jednu podrutu na 15 minuta srednje težine, alternativnu podrutu na 22 minute sa minimalnom teškoćom, i tako dalje.

Savetnik bira najefikasniju (najnižu cenu) ukupnu rutu na osnovu ciljeva koje su odredili korisnici i dostupnih statistika o putevima i stanju u saobraćaju.

### Komponente optimizatora

Optimizator sadrži tri komponente: transformator, procenjivač i generator plana. Sledeća slika ilustruje komponente.



Slika 8: Komponente optimizatora [5]

Skup blokova upita predstavlja raščlanjeni upit, koji je ulaz za optimizator. Operacije optimizatora prikazane su u tabeli 1:

|  |  |  |
| --- | --- | --- |
| Faza | Operacija | Opis |
| 1. | Transformacija upita (eng. Query Transformation ) | Optimizator utvrđuje da li je korisno promeniti oblik upita tako da optimizator može da generiše bolji plan izvršenja. |
| 2. | Procenjivač (eng. Estimation) | Optimizator procenjuje troškove svakog plana na osnovu statistike u rečniku podataka. |
| 3. | Generisanje plana (Plan Generation) | Optimizator upoređuje troškove planova i bira plan sa najnižom cenom, poznat kao plan izvršenja, koji će preći na generator izvora reda. |

Tabela 1: Komponente optimizatora [5]

#### Transformacija upita (Query Transformation)

Za neke izraze, transformator upita određuje da li je korisno prepisati originalni SQL izraz u semantički ekvivalentni SQL izraz sa nižom cenom.

Kada postoji održiva alternativa, baza podataka odvojeno izračunava trošak alternativa i bira alternativu sa najnižim troškovima. Transformator upita na primer često generiše plan koji prepisuje ulazni upit koji koristi ILI u izlazni upit koji koristi UNION ALL. [5]

#### Procenjivač (Estimation)

Procenjivač je komponenta optimizatora koja određuje ukupne troškove datog plana izvršenja. Koristi tri različite mere za određivanje troškova: selektivnost, kardinalnost i troškove.

1. ***Selektivnost***:

Selektivnost predstavlja procenat redova u skupu redova koje upit bira, pri čemu 0 znači da nema redova, a 1 znači da su izabrani svi redovi. Selektivnost je vezana za predikat upita, kao što je WHERE prezime LIKE 'A%', ili kombinaciju predikata. Predikat postaje selektivniji kako se vrednost selektivnosti približava 0, a manje selektivan (ili neselektivan) kako se vrednost približava 1.

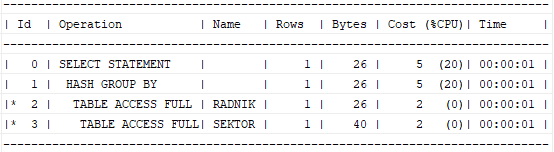
1. ***Kardinalnost:***

Kardinalnost je broj redova koje je svaka operacija vratila u planu izvršenja. Ovaj ulaz, koji je presudan za dobijanje optimalnog plana, zajednički je za sve funkcije troškova.

1. ***Trošak:***

Ova mera predstavlja jedinice rada ili resursa koji se koriste. Optimizator upita koristi I/O diska, upotrebu procesora i upotrebu memorije kao radne jedinice.

Dostupni statistički podaci omogućavaju procenjivaču da ih koristi za izračunavanje mera. Statistika poboljšava stepen tačnosti mera.

******

Slika 9: Prikaz statistike procenjivača

##### ***Selektivnost***

Skup redova može biti osnovna tabela, prikaz ili rezultat spajanja. Selektivnost je vezana za predikat upita, kao što je last\_name = 'Smith' ili kombinaciju predikata, kao što je last\_name = 'Smith' AND job\_id = 'SH\_CLERK'. Predikat filtrira određeni broj redova iz skupa redova. Selektivnost predikata prema tome ukazuje na to koliko redova prolazi test predikata. Selektivnost se kreće od 0.0 do 1.0. Vrednost od 0.0 znači da iz niza redova nisu izabrani redovi, dok selektivnost od 1,0 znači da su izabrani svi redovi. Predikat postaje sve selektivniji kako se vrednost približava 0.0, a manje selektivan (ili ne selektivan) kako se vrednost približava 1.0. [5]

Optimizator procenjuje selektivnost u zavisnosti od toga da li je statistika dostupna:

1. ***Statistika nije dostupna:***

U zavisnosti od vrednosti parametra OPTIMIZER\_DYNAMIC\_SAMPLING pokretanja, optimizator koristi dinamičku statistiku ili internu podrazumevanu vrednost. Baza podataka koristi različite interne podrazumevane vrednosti u zavisnosti od tipa predikata. *Primer: unutrašnja podrazumevana vrednost predikata jednakosti (last\_name = 'Smith') je manja nego za predikat opsega (last\_name > 'Smith') jer se očekuje da predikat jednakosti vrati manji deo redova*.

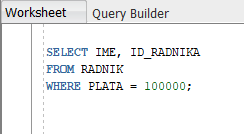
1. ***Statistika dostupna:***

Kada je statistika dostupna, estimator ih koristi za procenu selectivnosti. *Primer: Pretpostavimo da postoji 150 različitih prezimena zaposlenih. Za predikat jednakosti last\_name = 'Smith', selectivity je recipročna brouj n različitih vrednosti last\_name.*

##### ***Kardinalnost***

Kardinalnost je broj redova koje svaka operacija vraća u planu izvršenja. Ako je procena optimizatora za broj redova vraćenih skeniranjem cele tabele npr.100, onda je procena kardinalnosti za ovu operaciju 100. Procena kardinalnosti se beleži u koloni "Redovi" plana izvršenja. Optimizator određuje kardinalnost za svaku operaciju na osnovu složenog skupa formula koje kao unos koriste i statistiku .

Optimizator koristi neke od najjednostavnijih formula kada se u upitu za jednu tabelu pojavi jedan predikat jednakosti. U ovom slučaju, optimizator preuzima ujednačenu raspodelu i izračunava kardinalnost upita tako što deli ukupan broj redova u tabeli brojem različitih vrednosti u koloni koja je korišćena u predikatu odredbe WHERE. [5] *Primer: korisnik izvršava upit nad tabelom radnik na sledeći način:*



Slika 10: Primer upita

Primer: Tabela "Radnik" sadrži 107 redova. Trenutna statistika baze podataka pokazuje da je broj različitih vrednosti u koloni plata 58. Zbog toga optimizator procenjuje kardinalnost rezultata kao 2, koristeći formulu 107/58=1.84. Procene kardinalnosti moraju biti što preciznije jer utiču na sve aspekte plana izvršenja. Kardinalnost je takođe važna kada optimizator određuje cenu spoja. Na primer, u spajanju tabela zaposlenih i odeljenja, broj redova kod zaposlenih određuje koliko često baza podataka mora da ispita tabelu odeljenja.

##### ***Trošak***

Model troškova optimizatora uzima u obzir resurse mašine za koje se predviđa da će se koristiti upitom. Trošak je interna numerička mera koja predstavlja procenjenu upotrebu resursa za plan. Cena je specifična za upit u okruženju optimizatora. Da bi procenio troškove, optimizator uzima u obzir sledeće faktore: [5]

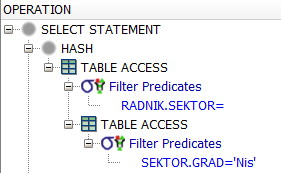
* Sistemski resursi, koji uključuju procenjeni I/O, CPU i memoriju
* Procenjeni broj vraćenih redova (kardinalnost)
* Veličina početnih skupova podataka
* Distribucija podataka
* Pristupne strukture

## *SQL Row Source Generation*

Generator izvora reda dobija najoptimalniji plan izvršenja koji mu prosleđuje komponente Generator plana (treća komponenta Optimizatora). Generator izvora reda je softver koji od dobijenog optimalnog plana izvršenje proizvodi iterativni plan, nazvan plan upita, koji je upotrebljiv od strane baze podataka. Iterativni plan je binarni program koji, kada ga izvrši SQL virtuelna mašina, daje skup rezultata. Plan upita predstavlja kombinaciju koraka. Svaki korak vraća skup redova. Redovi u ovom skupu se koriste u sledećim korakom ili se u poslednjem koraku vraćaju aplikaciji. [2]

Generator reda stvara stablo izvora reda, koje je kolekcija izvora reda. Stablo izvora reda prikazuje sledeće informacije: [2]

* Redosled tabela na koje se poziva upit
* Način pristupa za svaku tabelu navedenu u upitu
* Metoda pridruživanja za tabele na koje utiču operacije spajanja u upitu
* Operacije podataka kao što su filtriranje, sortiranje ili agregiranje



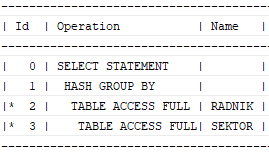
Slika 11: Grafički prikaz plana izvršenja upita

Izlaz iz generatora je plan izvršenja upita.

## SQL Izvršenje upita

Plan izvršenja predstavlja tačan redosled koraka koje je baza podataka preduzela za pokretanje upita, uključujući ključne parametre (koji pomažu u procesu podešavanja), koliko redova je svaka operacija vratila, i /O potreban za njenu obradu i koliko je vremena trebalo za izvršenje. Kada se šalje upit u bazu podataka, posao SQL optimizatora je da shvati kako da ga izvrši, a taj proces kreira plan izvršenja upita. [5]

*Primer, upita:*

**

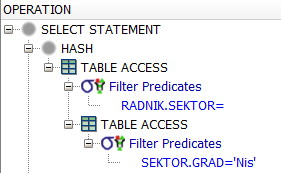
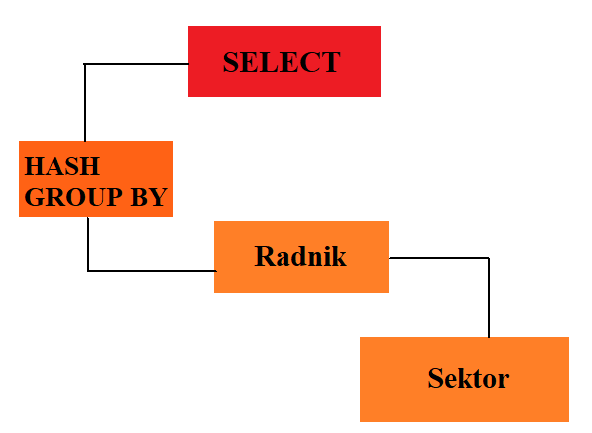
Slika 12: Tekstualni prikaz plana izvršenja

Svaka linija u planu je zasebna operacija. Ove operacije povezane su odnosom roditelja i deteta. Izraz SELECT je na vrhu odnosno predstavlja koren, tabele su listovi na dnu, a između njih se nalaze moguće operacije. Operacije se svrstavaju u tri kategorije: [6]

* Single-child operations – Operacije sa jednim detetom
* Multichild operations – Operacije sa jednim detetom
* Joins

Operacija sa jednim detetom uvek ima tačno jednu operaciju ispod sebe na stablu plana izvršenja. Najzastupljeniji primeri za to su koraci grupisanja i sortiranja. Multichild operacije, koje mogu imati jednu ili više operacija ispod sebe su retke, obično će to biti UNION operacija. Spojevi uvek imaju tačno dvoje dece. Svako od ove dece može sam da bude spoj, tabela ili bilo koja druga operacija plana. Čitanje plana izvršenja je jednostavno. U tekstualnom planu izvršenja, uvlačenje ukazuje na odnos roditelja/deteta. Roditelj svakog koraka je prva linija iznad njega, uvučena levo. Deca operacije su ona ispod nje, uvučena desno, do sledeće operacije na istoj dubini. [6]

Gornji plan se takodje može predstaviti preko grafa. Baza podataka koristi pretragu po dubini kako bi sledila plan izvršenja . Počinje od vrha stable i ide do lista. Potraga zatim ponavlja postupak, spuštajući se niz drvo do sledećeg lista, ponavlja se isti postupak dok se ne obiđu svi koraci u planu izvršenja.



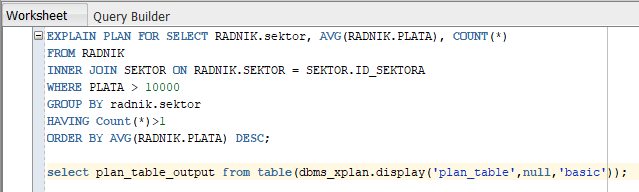
Slika 13: Plan izvršenja prikazan u vidu grafa

*Primer: Rredosled operacija za plan izvršenja gornjeg upita za dve tabele je sledeći:*

* *Obilazak stable počinje od vrha stabla (SELECT STATEMENT) , zatim se spušta niz drvo do prvog lista. List predstavlja TABELA - tabele SEKTOR.*
* *Redovi koji se dobijaju kao rezultat izvršenja date operacije iz ove tabele se prenose do roditelja prvog lista..*
* *Obilazi se sledeći korak koji predstavlja TABELA - tabele RADNIK.*
* *Redovi koji se dobijaju kao rezultat izvršenja date operacije iz ove tabele se prenose do roditelja, HASH GROUP BY.*
* *Pristupilo se svoj deci, izvršava se spajanje tabele a redovi koji predstavljaju rezultat prenose se u SELECT STATEMENT i nazad klijentu.*

**Explain plan** prikazuje planove izvršenja koje je izabrao Oracle optimizator za naredbe SELECT, UPDATE, INSERT i DELETE. Plan izvršenja upita je redosled operacija koje Oracle izvodi za pokretanje izraza. Stablo izvora reda predstavlja jezgro plana izvršenja. Prilikom korišćenja ove komande omogućava se pregled izvršenja upita pri čemu je moguće pre primene samog upita videti kojim redosledom će se izvršiti operacije i kakav rezultat se postiže upitom. Poredjenjem performansi (cena, broj kolone koje se obrađuju, itd.) može se odrediti da li upit zadovoljava potrebe ili ga je potrebno modifikovati kako bi postigli bolje rezultate. [7]

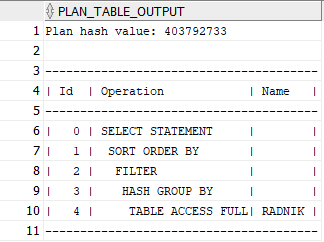
*Sintaksa :* ***EXPLAIN PLAN FOR*** *command;[8]*

**

Slika 14: Primena komande EXPLAIN PLAN

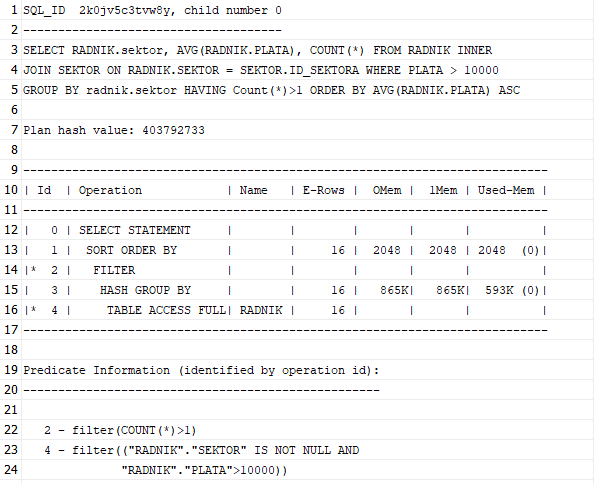
*Rezultat komand* **select** **plan\_table\_output frotable(dbms\_xplan.display('plan\_table',null,'basic'));[8]**

* PLAN\_TABLE se automatski kreira kao globalna privremena tabela koja sadrži izlaz EKSPLAIN PLAN

**

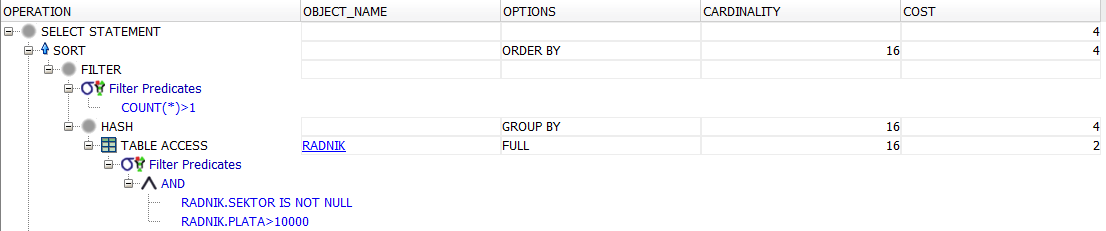
Slika 15: Rezultat komande EXPLAIN PLAN

*Rezultat komande* ***select \* from table(dbms\_xplan.display\_cursor(sql\_id=>'2k0jv5c3tvw8y', format=>'ALLSTATS LAST')); [8]***

******

Slika 16:Rezultat komande dbms\_xplan.display\_cursor

*Rezultat komande* ***V$SQL\_PLAN.SQL\_ID***:[8]

**

Slika 17: Rezultat komande V$SQL\_PLAN.SQL\_ID

# Zaključak

Najkritičnija funkcionalnost zahteva DBMS-a je sposobnost pravovremene obrade upita. Vreme potrebno za izvršenje upita i optimalnost upita delom zavise od načina obrade upita.

Na osnovu poglavlja koja su predstavljena u ovom radu može se zaključiti da se obrada upita sastoji od 3 osnovna koraka:

1. Parsiranje upita
2. Optimizator upita
3. Izvršenja upita

Parsiranje upita prihvata izjavu niskog nivoa, razdvaja tokene i proverava sintaksnu i semantičku ispravnost. Kada izjava prođe testiranje generise se plan upita u vidu tabele ili grafa koji se prosleđuje Optimizatoru. Optimizator se sastoji od 3 komponente: transformator upita, estimator i generator plana. Uloga Optimizatora je da generiše moguće planove izvršenja za zadati upit, odredi cenu svakog upita i izabere najoptimalniji upit na osnovu procenjenih parametara. Plan izvršenja u vidu grafa-tabele , prenosi se dalje na izvršenje. Komponenta koja je zadužena za izvršenje upita obilazi graf po dubini i izvršava operacije redom kojim ih posećuje unutar grafa.

Mehanizam obrade upita pored toga što nudi transformaciju upita i izbor najboljeg rešenja, omogućava korišćenje komandi za prikaz plana izvršenja pri čemu se može videti kako je optimizer izvršio upit i transformisao ga. Komanda kao što je EPLAIN PLAN omogućava tabelarni prikaz hijararhije izvršenja operacija, što korisniku pruža mogućnost da uvidi mogućnosti transformacije upita koje će dovesti do optimalnijeg izvršenja.

# Reference

# [1] Oracle Database Online Documentation 10*g* Release 2 (10.2), Introduction to the Oracle Database, link: <https://docs.oracle.com/cd/B19306_01/server.102/b14220/intro.htm>

[2] Oracle® Database, 11*g* Release 2 (11.2), link:<https://docs.oracle.com/cd/E11882_01/server.112/e40540.pdf> , May 2015

[3] Oracle Database Online Documentation 12*c* Release 1 (12.1), SQL Processing, link:<https://docs.oracle.com/database/121/TGSQL/tgsql_sqlproc.htm#TGSQL175>

# [4] Oracle Database Online Documentation 12*c* Release 1 (12.1), Tuning the Shared Pool and the Large Pool, link:<https://docs.oracle.com/database/121/TGDBA/tune_shared_pool.htm#TGDBA558>

# [5] Oracle Database Online Documentation 12*c* Release 1 (12.1), Query Optimizer Concepts,

link:<https://docs.oracle.com/database/121/TGSQL/tgsql_optcncpt.htm#TGSQL192>

# [6] Oracle Magazine, How to Read an Execution Plan, link: <https://blogs.oracle.com/oraclemagazine/how-to-read-an-execution-plan>

# [7] Oracle Database Online Documentation 10*g* Release 2 (10.2), Using EXPLAIN PLAN, link:<https://docs.oracle.com/cd/B19306_01/server.102/b14211/ex_plan.htm#g42231>

# [8] How do I display and read the execution plans for a SQL statement,

Link: <https://blogs.oracle.com/optimizer/how-do-i-display-and-read-the-execution-plans-for-a-sql-statement>