Elektronski fakultet, Niš

Smer: Računarstvo i informatika, Softversko inženjerstvo

Predmet: **Sistemi za upravljanje bazama podataka**

Tema:

Interna statistika koju Oracle baza podataka održava

Student: Lena Petrović 1295

Mentor: Doc. dr Aleksandar Stanimirović

Niš, April 2022

Statistika kod baza podataka

Statistika kod baza podataka je usko povezana sa optimizatorom, pa će s toga prvo biti par reči o tome.

Optimizator je "srce i duša" relacionog DBMS-a. On analizira SQL upite i određuje najefikasniji plan izvršenja naredbe. Proces analize se svodi na raščlanjivanje SQL naredbe da bi se odredilo kojim se tabelama i kolonama treba pristupiti. Nakon toga se analiziraju sistemske informacije i statistika koja se čuva u rečniku podataka kako bi se dobio najbolji plan izvršenja.

Optimizator se takođe može definisati kao ekspertski sistem za pristup podacima baze podataka. Ekspertski sistem je skup standardnih pravila koja u kombinaciji sa statističkim podacima mogu dati dobre rezultate (na primer, medicinski ekspertski sistem uzima skup pravila koja određuju koji je lek koristan za koju bolest, kombinuje ga sa podacima koji opisuju simptome bolesti i primenjuje tu bazu znanja na listu ulaznih simptoma).

Funkcija optimizatora se može shvatiti kao proces koji obuhvata četiri koraka:

1. Primanje i provera sintakse SQL naredbe
2. Analiziranje okruženja i optimizacija metoda za zadovoljavanje SQL naredbe
3. Kreiranje mašinski čitljivog uputstva za izvršavanje optimizovanog SQL-a
4. Izvršenje uputstva ili čuvanje za buduće izvršenje

Što se tiče koraka broj 2, postavlja se pitanje kako optimizator odlučuje kako da izvrši ogroman niz SQL naredbi koje se mogu poslati?

Optimizator ima mnogo strategija za optimizaciju SQL-a. Kako bira koju od ovih strategija da koristi u optimizovanim pristupnim putanjama? Podaci o unutrašnjem radu svojih optimizatora, vlasnici DBMS-ova nisu javno objavili, ali se zna da će dobar optimizator biti zasnovan na troškovima. Ovo znači da će optimizator uvek pokušati da formuliše pristupnu putanju za svaki upit tako da smanjuje ukupne troškove. Da bi ovo postigao, optimizator će primeniti formule troškova upita koje procenjuju i odmeravaju više faktora za svaki potencijalni pristupni put, kao što su CPU troškovi, I/O troškovi, statističke informacije…

Bez statistike, optimizatoru bi bilo teško da optimizuje bilo šta. Ove statistike pružaju optimizatoru informacije o stanju tabela kojima će pristupiti SQL izraz koji se optimizuje. Statistika je od vitalnog značaja za performanse upita. Bez njih, optimizator samo nagađa koja će permutacija puteva u podacima biti najefikasnija. Svaki pristup svakoj tabeli ne postaje ništa bolji od skeniranja cele tabele.

Statistika kod Oracle-a

Kada je Oracle baza podataka prvi put predstavljena, odluku o tome kako će se izvršavati SQL naredba određivao je optimizator zasnovan na pravilima (RBO - Rule Based Optimizer). Optimizator zasnovan na pravilima, kao što mu i ime kaže, prati skup pravila za određivanje plana izvršenja za SQL naredbe. Pravila su rangirana tako da ako postoje dva moguća pravila koja bi se mogla primeniti na SQL naredbu, koristilo bi se pravilo sa nižim rangom.

Kasnije je u Oracle 7 bazi podataka uveden optimizator zasnovan na troškovima (CBO - Cost Based Optimizator). Uveden je kako bi se bavio poboljšanom funkcionalnošću koja je dodata Oracle bazi podataka, a koja podrazumeva paralelno izvršavanje i particionisanje, i da bi se u obzir uzeo stvarni sadržaj i distribucija podataka. Optimizator zasnovan na trošku ispituje sve moguće planove za SQL naredbu i bira onaj sa najnižom cenom (cena predstavlja procenjenu upotrebu resursa za dati plan). Što je trošak niži, očekuje se da će plan izvršenja biti efikasniji. Da bi optimizator zasnovan na troškovima tačno odredio cenu za plan izvršenja mora da ima informacije o svim objektima (tabelama i indeksima) kojima se pristupa u SQL naredbi, kao i informacije o sistemu na kome će se SQL naredba pokrenuti.

Ove informacije koje su potrebne optimizatoru, nazivaju se statistike optimizatora. Razumevanje i upravljanje statistikom optimizatora je ključ za optimalno izvršavanje SQL naredbi. Znati kada i kako prikupiti statistiku na vreme je ključno.

Pristup optimizacije zasnovan na troškovima koristi statistiku za izračunavanje selektivnosti predikata i za procenu troškova svakog plana izvršenja. Selektivnost je deo redova u tabeli koji zadovoljavaju predikat SQL naredbe. Optimizator koristi selektivnost predikata da proceni cenu određenog metoda pristupa i da odredi optimalni redosled spajanja.

Statistika određuje distribuciju podataka i karakteristike skladištenja tabela, kolona, indeksa i particija. Optimizator koristi ove statistike da proceni koliko je ulazno-izlaznih podataka i memorije potrebno da se izvrši SQL naredba korišćenjem određenog plana izvršenja. Statistika se čuva u rečniku podataka (data dictionary) i može da se izveze iz jedne baze podataka i uveze u drugu (na primer, da bi se prenela statistika proizvodnje u sistem za testiranje radi simulacije stvarnog okruženja).

Statistika se mora redovno prikupljati da bi se optimizatoru pružile informacije o objektima šeme. Novu statistiku treba prikupiti nakon što su podaci ili struktura objekta šeme modifikovani na način koji prethodnu statistiku čini netačnom. Na primer, nakon učitavanja značajnog broja redova u tabelu, trebalo bi da se prikupi nova statistika o broju redova.

Generisani statistički podaci uključuju sledeće:

- Statistika tabela
    - Broj redova
    - Broj blokova
    - Broj praznih blokova
    - Prosečna dužina reda
- Statistika kolone
    - Broj različitih vrednosti u koloni
    - Broj nula u koloni
    - Distribucija podataka (histogram)
- Statistika indeksa
    - Broj blokova listova
    - Nivoi
    - Faktor grupisanja
- Statistika sistema
    - I/O performanse i korišćenje
    - Performanse i korišćenje procesora

Generisanje statistike

Pošto se pristup zasnovan na troškovima oslanja na statistiku, trebalo bi da se generiše statistika za sve tabele i klastere i sve tipove indeksa kojima pristupaju SQL naredbe pre korišćenja pristupa zasnovanog na troškovima. Ako se veličina i distribucija podataka u tabelama često menjaju, trebalo bi redovno da se generišu ove statistike kako bi bili sigurni da statistika tačno predstavlja podatke u tabelama.

Oracle generiše statistiku koristeći sledeće tehnike:

- Procena na osnovu slučajnog uzorkovanja podataka
- Tačno izračunavanje
- Korisnički definisane metode prikupljanja statistike

Da bi tačno izračunao, Oracle-u je potrebno dovoljno prostora za skeniranje i sortiranje tabele. U slučaju da u memoriji nema dovoljno prostora, možda će biti potreban privremeni prostor. Za procene, Oracle-u je potrebno dovoljno prostora da izvrši skeniranje i sortira samo redove u traženom uzorku tabele. Neki statistički podaci se uvek tačno izračunavaju (na primer, broj blokova podataka koji trenutno sadrže podatke u tabeli). Preporučljivo je koristiti procenu za tabele i klastere umesto za izračunavanje, osim ako nisu potrebne tačne vrednosti. Pošto se procena retko sortira, često je mnogo brža od izračunavanja, posebno za velike tabele. Da bi

procenio statistiku, Oracle bira nasumični uzorak podataka. Može se odrediti procenat uzorkovanja i da li uzorkovanje treba da se zasniva na redovima ili blokovima.

Uzorkovanje redova čita redove bez obzira na njihov fizički položaj na disku. Ovo daje najviše nasumičnih podataka za procene, ali može se desiti da je pročitano više podataka nego što je potrebno. Na primer, u najgorem slučaju uzorak reda može izabrati jedan red iz svakog bloka, što se svodi na potpuno skeniranje tabele ili indeksa.

Uzorkovanje blokova čita nasumični uzorak blokova i koristi sve redove u tim blokovima kako bi izračunao procenu. Ovo smanjuje količinu ulazno-izlaznih aktivnosti za datu veličinu uzorka, ali može smanjiti slučajnost uzorka ako redovi nisu nasumično raspoređeni na disku. Uzorkovanje blokova nije dostupno za statistiku indeksa.

Kod generisanja statistike za tabelu, kolonu ili indeks, ako rečnik podataka već sadrži statistiku za objekat, Oracle ažurira postojeću statistiku.

Sledeći put kada se ista SQL naredba izvršava, optimizator automatski bira novi plan izvršenja na osnovu nove statistike. Distribuirani iskazi izdati na udaljenim bazama podataka koje pristupaju analiziranim objektima koriste novu statistiku sledeći put kada ih Oracle analizira.

Objekti particionirane šeme mogu sadržati više različitih statistika. Oni mogu imati statistiku koja se odnosi na ceo objekat šeme u celini (globalna statistika), mogu imati statistiku koja se odnosi na pojedinačnu particiju, i mogu imati statistiku koja se odnosi na pojedinačnu podparticiju kompozitnog particionisanog objekta.

Moguće je generisanje globalne statistike iz statistike na nivou particije, međutim nije uvek jednostavno. Na primer, veoma je teško odrediti broj različitih vrednosti za kolonu iz broja različitih vrednosti koje se nalaze u svakoj particiji zbog mogućeg preklapanja vrednosti. Zbog toga se preporučuje prikupljanje globalne statistike uz pomoć paketa DBMS_STATS, umesto ANALYZE.

Statistika se može generisati naredbom ANALYZE ili paketom DBMS_STATS. DBMS_STATS je preferirani metod za prikupljanje statistike i zamenjuje sada već zastarelu ANALYZE komandu.

ANALYZE naredba

ANALYZE naredba može da generiše statistiku za optimizaciju zasnovanu na troškovima. Međutim, korišćenje Analyze u ovu svrhu se ne preporučuje zbog raznih ograničenja, na primer:

- Serijsko izvršenje
- Računa se globalna statistika za particionisane tabele i indekse umesto da se direktno prikuplja. Ovo može dovesti do netačnosti za neke statistike (broj različitih vrednosti).
    - Za particionirane tabele i indekse, Analyze prikuplja statistiku za pojedinačne particije i zatim izračunava globalnu statistiku iz statistike particije.
    - Za kompozitno particionisanje, Analyze prikuplja statistiku za podparticije, a zatim izračunava statistiku particije i globalnu statistiku iz statistike podparticija.
- Analyze naredba ne može prepisati ili izbrisati neke od vrednosti statistike koje je prikupio DBMS_STATS

Primeri sintakse Analyze naredbe mogu se videti na slici ispod. Priloženi su primer analiziranja table i indeksa.

```
ANALYZE table scott compute statistics;
ANALYZE table scott estimate statistics sample 25 percent;
ANALYZE table scott estimate statistics sample 1000 rows;
analyze index sc_idx compute statistics;
analyze index sc_idx validate structure;
```

DBMS_STATS paket

Paket DBMS_STATS omogućava generisanje i upravljanje statistikom za optimizaciju zasnovanu na troškovima. Može se koristiti za prikupljanje, modifikovanje, pregled i brisanje statistike. Takođe se može koristiti za čuvanje skupova statističkih podataka. DBMS_STATS paket može prikupiti statistiku o indeksima, tabelama, kolonama i particijama, kao i statistiku o svim objektima šeme u šemi ili bazi podataka. Ne prikuplja statistiku klastera (može se koristiti DBMS_STATS za prikupljanje statistike o pojedinačnim tabelama umesto o celom klasteru)

Operacije prikupljanja statistike mogu se odvijati ili serijski ili paralelno. Kad god je to moguće, DBMS_STATS poziva paralelni upit da prikupi statistiku sa navedenim stepenom paralelizma (u suprotnom, poziva serijski upit ili Analyze naredba). Indeksna statistika se ne prikuplja paralelno.

Za particionisane tabele i indekse, DBMS_STATS može prikupiti zasebne statistike za svaku particiju, kao i globalnu statistiku za celu tabelu ili indeks. Slično, za kompozitno particionisanje DBMS_STATS može da prikupi odvojene statistike za podparticije, particije i celu tabelu ili indeks. U zavisnosti od SQL naredbe koja se optimizuje, optimizator može izabrati da koristi ili statistiku particije (ili podparticije) ili globalnu statistiku.

DBMS_STATS prikuplja statistiku samo za optimizaciju zasnovanu na troškovima, ne prikuplja druge statistike. Na primer, statistika tabele koju prikuplja DBMS_STATS uključuje

broj redova, broj blokova koji trenutno sadrže podatke i prosečnu dužinu reda, ali ne i broj ulančanih redova, prosečan slobodan prostor ili broj neiskorišćenih blokova podataka.

Lista procedura za prikupljanje statistika:

Procedure	Description
`GATHER_INDEX_STATS`	Collects index statistics.
`GATHER_TABLE_STATS`	Collects table, column, and index statistics.
`GATHER_SCHEMA_STATS`	Collects statistics for all objects in a schema.
`GATHER_DATABASE_STATS`	Collects statistics for all objects in a database.

Za objekte baze podataka koji se stalno menjaju, statistika se mora redovno prikupljati tako da tačno opisuje objekat baze podataka. DBMS_STATS paket sadrži preko 50 različitih procedura za prikupljanje i upravljanje statistikom. Za najvažnije od ovih procedura smatraju se GATHER_*_STATS procedure.

Primer procedure

Procedura DBMS_STATS.GATHER_TABLE_STATS omogućava prikupljanje statistike tabela, particija, indeksa i kolona. Njoj je potrebno 15 različitih parametara, ali samo prva dva moraju biti specificirana da bi se pokrenula procedura, to su:

- Naziv šeme koja sadrži tabelu
- Naziv tabele

Parametri svih procedura iz GATHER_*_STATS dela su slični, pa će ovde biti nabrojani parametri GATHER_TABLE_STATS procedure:

- estimate_percent
- block_sample
- method_opt
- degree
- granularity
- cascade
- stattab
- static
- options

- objlist
- statown
- gather_sys
- no_invalidate

Tabele statistika

Paket DBMS_STATS omogućava čuvanje statistike u *tabeli statistike*. U tabelu se može uneti statistika za kolonu, tabelu, indeks ili šemu i zatim vratiti te statistike u rečnik podataka (data dictionary). Optimizator ne koristi statistiku koja se čuva u tabeli statistike.

Tabela statistike omogućava eksperimentisanje sa različitim skupovima statistika. Na primer, pre brisanja određenog skupa statističkih podataka, može se napraviti rezervna kopija, ili se mogu uporediti performanse SQL naredbe optiminizovane različitim skupovima statistika (pa se nakon toga u rečnik podataka upisuju statistike sa najboljim performansama).

Tabela statistike može da čuva više različitih skupova statistike ili se može kreirati više tabela statistike da bi se odvojeno čuvali različiti skupovi statistike.

Prikupljanje statistike

Sledeće procedure paketa DBMS_STATS koriste se za prikupljanje statistike određenog tipa:

- GATHER_DATABASE_STATS
- GATHER_DICTIONARY_STATS
- GATHER_FIXED_OBJECTS_STATS
- GATHER_INDEX_STATS
- GATHER_SCHEMA_STATS
- GATHER_SYSTEM_STATS
- GATHER_TABLE_STATS

Skladištenje i preuzimanje statistike

Sledeće procedure paketa DBMS_STATS koriste se za skladištenje i preuzimanje pojedinačnih statističkih podataka vezanih za kolone, indekse i tabele:

- SET_COLUMN_STATS
- SET_INDEX_STATS
- SET_SYSTEM_STATS
- SET_TABLE_STATS
- GET_COLUMN_STATS
- GET_INDEX_STATS
- GET_SYSTEM_STATS

- GET_TABLE_STATS

Što se tiče ove grupe procedura, neophodno je napomenuti da se se koristi za testiranje sistema, da se sistem ‘prevari’ (kako bi se pratilo ponašanje optimizatora kada misli da kolona ima drugačije karakteristike). Moguće je podesiti podatke kao što su broj različitih vrednosti i broj null vrednosti kod kolona ili broj redova i njegova prosečna dužina kod tabela.

Brisanje statistike

Sledeće procedure paketa DBMS_STATS koriste se za brisanje statistika:

- DELETE_COLUMN_STATS
- DELETE_DATABASE_STATS
- DELETE_DICTIONARY_STATS
- DELETE_FIXED_OBJECTS_STATS
- DELETE_INDEX_STATS
- DELETE_SCHEMA_STATS
- DELETE_SYSTEM_STATS
- DELETE_TABLE_STATS

Prenos statistike

- EXPORT_COLUMN_STATS
- EXPORT_DATABASE_STATS
- EXPORT_DICTIONARY_STATS
- EXPORT_FIXED_OBJECTS_STATS
- EXPORT_INDEX_STATS
- EXPORT_SCHEMA_STATS
- EXPORT_SYSTEM_STATS
- EXPORT_TABLE_STATS
- IMPORT_COLUMN_STATS
- IMPORT_DATABASE_STATS
- IMPORT_DICTIONARY_STATS
- IMPORT_FIXED_OBJECTS_STATS
- IMPORT_INDEX_STATS
- IMPORT_SCHEMA_STATS
- IMPORT_SYSTEM_STATS
- IMPORT_TABLE_STATS

Zaključavanje i otključavanje statistike

Sledeće procedure paketa DBMS_STATS koriste se za zaključavanje i otključavanje statistike o objektima:

- LOCK_SCHEMA_STATS
- LOCK_TABLE_STATS

- UNLOCK_SCHEMA_STATS
- UNLOCK_TABLE_STATS

Pregled statistike

Može se koristiti paket DBMS_STATS radi pregleda statistike skladištene u rečniku podataka ili u tabeli statistike.

Takođe se mogu koristiti sledeće funkcije rečnika podataka za statistiku u rečniku podataka:

- USER_TABLES, ALL_TABLES i DBA_TABLES
- USER_TAB_COLUMNS, ALL_TAB_COLUMNS i DBA_TAB_COLUMNS
- USER_INDEXES, ALL_INDEXES i DBA_INDEXES
- USER_CLUSTERS i DBA_CLUSTERS
- USER_TAB_PARTITIONS, ALL_TAB_PARTITIONS i DBA_TAB_PARTITIONS
- USER_TAB_SUBPARTITIONS, ALL_TAB_SUBPARTITIONS i DBA_TAB_SUBPARTITIONS
- USER_IND_PARTITIONS, ALL_IND_PARTITIONS i DBA_IND_PARTITIONS
- USER_IND_SUBPARTITIONS, ALL_IND_SUBPARTITIONS i DBA_IND_SUBPARTITIONS
- USER_PART_COL_STATISTICS, ALL_PART_COL_STATISTICS i DBA_PART_COL_STATISTICS
- USER_SUBPART_COL_STATISTICS, ALL_SUBPART_COL_STATISTICS i DBA_SUBPART_COL_STATISTICS

Razlika između ovih tabela po grupama (na primer USER_TABLES, ALL_TABLES i DBA_TABLES):

- USER_TABLES su tabele koje korisnik (user) poseduje
- ALL_TABLES su sve tabele kojima je odobren pristup
- DBA_TABLES su sve tabele u bazi podataka

Sva tri su prikazi osnovnih SYS tabela, ali prikazi USER_ i ALL_ pridružuju se korisničkom imenu (odnose se na prava pristupa, bezbednosne informacije…).

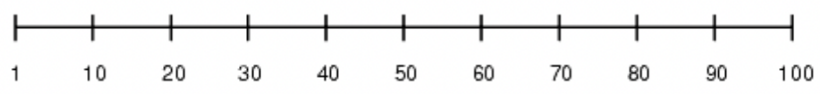
Histogrami

Optimizator zasnovan na troškovima koristi histograme vrednosti podataka da bi dobio tačne procene distribucije podataka kolone. Histogram deli vrednosti u koloni u trake, tako da sve vrednosti kolona u opsegu spadaju u isti opseg. Histogrami daju poboljšane procene selektivnosti u prisustvu iskrivljenih podataka (skew), što rezultira optimalnim planovima izvršenja.

Optimizator zasnovan na troškovima koristi histograme zasnovane na visini na određenim atributima da opiše distribuciju neuniformnih podataka. U histogramu zasnovanom na

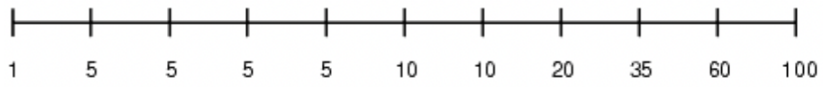
visini, vrednosti kolona su podeljene u trake tako da svaka traka sadrži približno isti broj vrednosti.

Ukoliko se posmatra jedna kolona sa vrednostima između 1 i 100 i histogram sa 10 segmenta, u slučaju da su podaci ravnomerno raspoređeni, histogram bi ovako izgledao:



Može se primetiti da su podaci uniformno rasporedjeni uzevši u obzir da četiri desetine podataka ima vrednost između 60 i 100.

Kada podaci nisu ravnomerno raspoređeni, histogram bi izgledao ovako:



Sada većina redova (podataka) ima vrednost 5 za posmatranu kolonu. U ovom primeru, samo jedna desetina podataka ima vrednosti između 60 i 100.

Histogrami mogu uticati na performanse i treba ih koristiti samo kada značajno poboljšavaju planove upita. Generalno, histogrami bi trebalo da se kreiraju za kolone koje se često koriste u where delu upita i imaju neravnomerno raspoređene podatke.

Proširena statistika

U Oracle 11g bazi podataka, uvedena su proširenja za statistiku kolona. Proširena statistika obuhvata dva dodatna tipa statistike: grupe kolona i statistiku izraza.

Grupe kolona

U podacima iz stvarnog sveta, često postoji veza između podataka koje se nalaze u različitim kolonama iste tabele. Koristeći samo osnovnu statistiku kolona, optimizator ne zna ništa o ovim realnim odnosima i mogao bi pogrešno da izračuna kardinalnost u slučaju da se više kolona iz iste tabele koristi u naredbi. Optimizator može da “postane svestan” ovih odnosa u stvarnom svetu tako što će imati proširenu statistiku o ovim kolonama kao grupi.

Primer: tabela Students koja sadrži u sebi kolone FacultyName i FacultyId koje su zavisne.

Kada god se u naredbi nađu kolone koje pripadaju nekoj grupi kolona, optimizator će koristiti statistiku grupe umesto statistike pojedinačnih kolona. Da bi se koristila statistika grupe kolona, nije neophodno da se sve kolone iz grupe pojavljuju u naredbi.

```
select dbms_stats.create_extended_stats('System', 'Students', '(FacultyName + FacultyId)') from dual;
```

	DBMS_STATS.CREATE_EXTENDED_STATS('SYSTEM','STUDENTS1','(FACULTYNAME+FACULTYID)')
1	SYS_STU$#VS4H26HNS02$N3D4EC0YG

Nakon kreiranja grupe kolona i sakupljanja statistika, može se videti da postoji dodatna kolona sa imenom koje je generisano od strane samog sistema. Dodatna kolona predstavlja statistiku grupe kolona i može se videti u USER_TAB_COL_STATISTICS tabeli.

Takođe, u tabeli USER_STAT_EXTENSIONS je moguće videti pregled ostalih postojećih kreiranih proširenih statistika.

```
select * from user_stat_extensions where creator = 'USER'
```

	TABLE_NAME	EXTENSION_NAME	EXTENSION	CREATOR	DROPPABLE
1	STUDENTS	SYS_STU$#VS4H26HNS02$N3D4EC0YG	(TO_NUMBER("FACULTYNAME")+TO_NUMBER("FACULTYID"))	USER	YES

Kada se kreira statistika o grupi kolona, optimizator može da izračuna bolju procenu kardinalnosti za kolone između kojih postoji veza. Može se koristiti funkcija DBMS_STATS.CREATE_EXTENDED_STATS da bi se definisala grupa kolona.

Statistika izraza

Moguće je kreirati proširenu statistiku za izraz, kako bi optimizatoru bilo lakše da se proceni kardinalnost kod “where” klauzule sa delom vezanim za neku kolonu. Ukoliko se često postavljaju upiti koji imaju isti “where” deo (na primer, često se vrednost neke numeričke kolone zaokružuje na dve decimale, ili se često povećava prvo slovo neke kolone), onda bi bilo korisno kreirati proširenu statistiku za izraz.

```
select dbms_stats.create_extended_stats('System', 'Students', '(UPPER(FULLNAME))') from dual;
```

	DBMS_STATS.CREATE_EXTENDED_STATS('SYSTEM','STUDENTS','(UPPER(FULLNAME))')
1	SYS_STUPRB$NKRGE_P68KQB5L4BP9K

Kao i kod grupe kolone, neophodno je ponovo pribaviti statistiku. Slično kao i kod grupe kolona, moguće se videti dodatnu kolonu u USER_TAB_COL_STATISTICS tabeli.

Kardinalnost

Kardinalnost je očekivani broj redova koji će biti vraćeni svakom operacijom. Jedna od najjednostavnijih formula za računanje kardinalnosti se koristi kada postoji jedna predikat jednakosti u jednom upitu tabele. U ovom slučaju optimizator preuzima uniformnu distribuciju i izračunava kardinalnost za upit deljenjem ukupnog broja redova u tabeli sa brojem različitih vrednosti u koloni koja se koristi u predikatu where dela naredbe.

```
explain plan for
select *
from Persons
where firstname = 'Lena';

select * from table(dbms_xplan.display);
```

Potrebno je izvršiti naredbe na slici iznad (prva kojom se kreira plan za dobavljanje svih redova kolone Persons gde kolona firstname ima vrednost 'Lena' i druga koja prikazuje plan), da bi se dobio plan izvršenja koji je prikazan na slici ispod.

	PLAN_TABLE_OUTPUT
1	Plan hash value: 2258600491
2	
3	--
4	| Id | Operation | Name | Rows | Bytes | Cost (%CPU)| Time |
5	--
6	| 0 | SELECT STATEMENT | | 1 | 31 | 2 (0)| 00:00:01 |
7	|* 1 | TABLE ACCESS FULL| PERSONS | 1 | 31 | 2 (0)| 00:00:01 |
8	--
9	
10	Predicate Information (identified by operation id):
11	---
12	
13	1 - filter("FIRSTNAME"='Lena')

Kolona firstaname ima 11 različitih vrednosti. Na osnovu gore rečenog, kardinalost za ovaj upit se izračunava kao količnik ukupnog broja redova i broja jedinstvenih vrednosti kolona. U ovom slučaju to bi bilo 13/11, što je 1,18, odnosno 1 red kada se zaokruži.

Važno je da procene kardinalnosti budu što tačnije jer utiču na sve aspekte plana izvršenja (od metode pristupa, do spajanja). Međutim, nekoliko faktora može dovesti do netačnih procena kardinalnosti čak i kada su osnovne statistike tabele i kolone tačne. Neki od ovih faktora uključuju:

- Iskrivljenost podataka
- Više predikata jedne kolone u jednoj tabeli
- Kolone umotane u funkciju u predikatima klauzule VHERE
- Složeni izrazi

Automatsko prikupljanje statistike

Oracle će automatski prikupljati statistiku za sve objekte baze podataka za koje ne postoji statistika ili imaju zastarelu statistiku. Ovo se može postići pokretanjem DBMS_STATS.GATHER_DATABASE_STATS_JOB_PROC procedure tokom unapred definisanog vremena (može se konkretno specificirati vremenski okvir za svaki dan ponaosob, ili za radne dane i vikend…). Ova procedura čita i posmatra sve parametre (koji su prethodno postavljeni uz pomoć SET_PARAM procedure), što znači da se mogu dobiti različite statistike.

Postavljanje parametara uz pomoć SET_PARAM procedure:

```
  1  begin
  2  DBMS_STATS.SET_PARAM('CASCADE','TRUE');
  3  DBMS_STATS.SET_PARAM('DEGREE','4');
  4  DBMS_STATS.SET_PARAM('METHOD_OPT','FOR ALL INDEXED COLUMNS SIZE 251');
  5  DBMS_STATS.SET_PARAM('ESTIMATE_PERCENT','8');
  6* end;
SQL> /
```

Kreiranje procedure koja će obavljati posao. Procedura sa slike omogućuje da se GATHER_STATS_JOB pokreće svakog prvog dana u mesecu, u 22:30 uveče. Primećuje se da se uzorkuje 8 procenata svake tabele za koju prikuplja statistika i da se kreiraju histogrami sa 251 krajnjom tačkom na svim indeksiranim kolonama.

```
SQL> begin
  2  dbms_scheduler.create_schedule(
  3      schedule_name => 'STATS_COLLECTION',
  4      repeat_interval=>'freq=monthly;bymonthday=1;byhour=22;byminute=30',
  5      comments => 'Schedule to collect statistics');
  6  dbms_scheduler.set_attribute(
  7      name=>'GATHER_STATS_JOB',
  8      attribute=>'SCHEDULE_NAME',
  9      value=>'STATS_COLLECTION');
 10  end;
```

Upravljanje statistikom

Pored prikupljanja statističkih podataka, podjednako je važno obezbediti uspešno upravljanje skupljenim statistikama. Kod Oracle-a postoje brojne metode za to. Neke od njih uključuju mogućnost vraćanja statistike na prethodnu verziju, opciju prenosa statistike sa jednog

sistema na drugi ili ručno podešavanje statističkih vrednosti. Ove opcije su korisne u određenim slučajevima, ali ne preporučuje se da zamene standardne metode prikupljanja statistike korišćenjem paketa DBMS_STATS.

Vraćanje statistike

Kada se statistika prikuplja uz pomoć DBMS_STATS paketa, originalna statistika se automatski čuva kao rezervna kopija u tabelama rečnika i može se lako vratiti pokretanjem DBMS_STATS.RESTORE_TABLE_STATS. Primer poziva procedure nakon koga je statistika vraćena statistiku koja je bila aktuelna u zadatom vremenu:

```
exec dbms_stats.restore_table_stats('System', 'Persons', to_timestamp('2022-04-14:10:50:30','yyyy-mm-dd:hh24:mi:ss'));
```

Statistika na čekanju

Kada se prikupe statistički podaci, oni se odmah upisuju u odgovarajuće tabele rečnika i optimizator počinje da ih koristi. U novijim verzijama Oracle baze, moguće je prikupiti statistiku optimizatora, ali ih ne objaviti odmah. Umesto toga se čuvaju u neobjavljenom, „na čekanju“, stanju. Umesto da se statistika upiše u tabele rečnika, one se čuvaju u tabelama čekanja kako bi se mogle testirati pre nego što budu objavljene. Ovo se rešava korišćenjem neke DBMS_STATS.SET_*_PREFS procedure u svrhu promene “publish” parametra. Podrazumevana vrednost je “true”, a potrebno je promeniti na “false”.

```
exec dbms_stats.set_table_prefs('System','Persons','publish','false');
```

Nakon ovoga, statistika se normalno prikuplja:

```
exec dbms_stats.gather_table_stats('System', 'Persons')
```

U tabeli USER_TAB_PENDING_STATS se nalaze sve statistike tabela koje su na čekanju.

```
select * from USER_TAB_PENDING_STATS
```

	TABLE_NAME	PARTITION_NAME	SUBPARTITION_NAME	NUM_ROWS	BLOCKS	AVG_ROW_LEN	IM_IMCU_COUNT	IM_BLOCK_COUNT	SCAN_RATE	SAMPLE_SIZE	LAST_ANALYZED
1	PERSONS	(null)	(null)	13	1	30	(null)	(null)	(null)	13	16-APR-22

Optmizatoru se može naznačiti da koristi statistiku koja je na čekanju u toku obrade upita, dok se procedurom PUBLISH_PENDING_STATS objavljuje navedena statistika koja je na čekanju.

Izvoz/uvoz statistike

Nalazi značajnu primenu kod testiranja. Kada je potrebno imati identičan sistem za testiranje kao i krajnji proizvod, kopiranjem statistike optimizatora iz jedne baze u drugu, može se videti ponašanje optimizatora. Procedure koje je potrebno iskoristiti za izvoz, a kasnije i uvoz podataka: DBMS_STATS.EXPORT_*_STATS i DBMS_STATS.IMPORT_*_STATS.

Upoređivanje statistika

Da bi se identifikovale razlike u statistikama, funkcije DBMS_STATS.DIFF_TABLE_STATS_* se koriste za upoređivanje statistike za tabelu iz dva različita izvora. Izvori statistike mogu biti: dve različite verzije podataka, statistika na čekanju i trenutna statistika, trenutna statistika i korisnička statistika…

Na sledećim slikama je moguće videti upoređivanje statistike iste tabele (“Persons”) u dva različita vremenska trenutka. Na prvoj slici se nalazi naredba koju je neophodno pokrenuti, a u kojoj figuriše DIFF_TABLE_STATS_IN_HISTORY paketa DBMS_STATS.

```
select * from table(dbms_stats.diff_table_stats_in_history(
                ownname => user,
                tabname => upper('&tabname'),
                time1 => systimestamp,
                time2 => to_timestamp('&time2','yyyy-mm-dd:hh24:mi:ss'),
                pctthreshold => 0));
```

Naredba zahteva unos naziva tabele za koje se upoređuje statistika. Jedno vreme je postavljeno na trenutno vreme sistema, dok se za drugo vreme unosi željeno vreme u prošlosti. Takođe postoji “pctthreshold” parametar koji je postavljen na 0. Ova inicijalizacija znači da će izveštaj upoređivanja biti generisan čak i u slučaju da ne postoji nikakva razlika u statistikama. Ukoliko je recimo “pctthreshold” parametar postavljen na 50, izveštaj će se generisati samo ako se statistike razlikuju za više od 50%. Rezultat se može videti na slici ispod:

```
STATISTICS DIFFERENCE REPORT FOR:
.................................

TABLE         : PERSONS
OWNER         : SYSTEM
SOURCE A      : Statistics as of 2022-04-15:12:45:36
SOURCE B      : Statistics as of 2022-04-14:12:35:59
PCTTHRESHOLD  : 0
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

TABLE / (SUB)PARTITION STATISTICS DIFFERENCE:
.............................................

OBJECTNAME                  TYP SRC ROWS       BLOCKS     ROWLEN     SAMPSIZE
.....................................................................................

PERSONS                     T   A   12         1          30         12
                                B   7          1          31         7
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

COLUMN STATISTICS DIFFERENCE:
.............................

COLUMN_NAME     SRC NDV     DENSITY    HIST NULLS   LEN  MIN    MAX    SAMPSIZ
.....................................................................................

ADDRESS         A   10      .1         NO   0       12   424B2  52656  12
                B   7       .142857142 NO   0       13   42756  52656  7
FIRSTNAME       A   11      .090909090 NO   0       6    416E6  56657  12
                B   6       .166666666 NO   0       5    416E6  56657  7
LASTNAME        A   12      .083333333 NO   0       6    416E6  56657  12
                B   7       .142857142 NO   0       6    416E6  56657  7
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

U izveštaju je moguće uočiti dve različite boje koje predstavljaju dva različita izvora statistika. Zelena boja, koja se odnosi na izvor A, označava statistiku vezanu za trenutno vreme. Crvena boja, koja se odnosi na izvor B, označava statistiku vezanu za neki vremenski trenutak u prošlosti. Nakon ovoga se može uočiti da uzorak A ima više redova od uzorka B (A - 12, B - 7 redova).

Zaključavanje statistike

U nekim slučajevima, može biti potrebno da se spreči prikupljanje novih statistika u tabeli ili šemi zaključavanjem statistike. Jednom kada se statistika zaključa, ne mogu se vršiti modifikacije te statistike sve dok se statistika ne otključa (izuzetak: parametar “FORCE” postavljen na “TRUE” kod GATHER_*_STATS).

Za sprečavanje skupljanja statistike iz određene tabele, zaključavanje je moguće izvršiti sledećom procedurom:

```
exec dbms_stats.lock_table_stats('System', 'Persons')
```

Dalje je moguće videti pokušaj prikupljanja statistike.

```
exec dbms_stats.gather_table_stats('System', 'Persons')
```

```
Error starting at line : 22 in command -
BEGIN dbms_stats.gather_table_stats('System', 'Persons'); END;
Error report -
ORA-20005: object statistics are locked (stattype = ALL)
```

"Prisiljeno" skupljanje statistike izgleda ovako:

```
exec dbms_stats.gather_table_stats('System', 'Persons', FORCE => TRUE)
```

```
PL/SQL procedure successfully completed.
```

Otključavanje tabele:

```
exec dbms_stats.unlock_table_stats('System', 'Persons')
```

Ostale statistike

Pored osnovne statistike tabela, kolona i indeksa, optimizator koristi i dodatne informacije da bi odredio plan izvršenja naredbe. Dodatne informacije podrazumevaju dinamičko uzorkovanje i sistemske statistike.

Dynamic sampling - Dinamičko uzorkovanje

Cilj dinamičkog uzorkovanja je povećanje postojeće statistike. Koristi se kada redovna statistika nije dovoljna za dobijanje kvalitetnih procena kardinalnosti.

Tokom sastavljanja SQL naredbe, optimizator odlučuje da li će koristiti dinamičko uzorkovanje ili ne, na osnovu toga da li su dostupne statistike dovoljne za generisanje dobrog plana izvršenja. Ako dostupna statistika nije dovoljna, koristiće se dinamičko uzorkovanje. Obično se koristi da se nadoknadi nedostajuća ili nedovoljna statistika (za koju se smatra da će loše uticati na plan izvršenja). Na primer, kada jedna ili više tabela u upitu nemaju statistiku, optimizator koristi dinamičko uzorkovanje da prikupi osnovne statistike o ovim tabelama pre optimizacije upita. Podaci o statistici prikupljeni u ovakvom slučaju nisu potpuni i visokog kvaliteta kao statistički podaci koji su skupljeni uz pomoć DBMS_STATS paketa.

Još jedan primer kada je potrebno dinamičko uzorkovanje je kada ne postoji proširena statistika. Na mestu kada bi trebalo da se koristi proširena statistika (na primer, u "where" delu upita figurišu dve kolone koje su zavisne), a ona ne postoji, neophodno je dinamičko uzorkovanje.

U novijim verzijama optimizator će automatski odlučiti da li će dinamičko uzorkovanje biti korisno i koji nivo dinamičkog uzorkovanja će se koristiti. Nivo dinamičkog uzorkovanja modifikuju se uz pomoć OPTIMIZER_DYNAMIC_SAMPLING parametra. Vrednost nivoa se kreće u opsegu 0-11 i kontroliše dve stvari. Kada će dinamičko uzorkovanje krenuti i koliko veliki skup uzoraka će biti korišćen za prikupljanje statistike. Što je veća veličina uzorka, duže je vremena potrebno.

Provera vrednosti OPTIMIZER_DYNAMIC_SAMPLING parametra:

```
show parameter optimizer_dynamic_sampling
```

```
NAME                          TYPE    VALUE
----------------------------- ------- -----
optimizer_dynamic_sampling integer 2
```

Tabela na kojoj je pokazano dinamičko uzorkovanje: tabela Students koja sadrži u sebi kolone FacultyName i FacultyId koje su zavisne, proširena statistika ne postoji.

```
explain plan for  select * from Students where FacultyName = 'Elektronski' and FacultyId = '1';
select * from table(dbms_Xplan.display);
```

```
PLAN_TABLE_OUTPUT
Plan hash value: 4078133427

-------------------------------------------------------------------------------
| Id  | Operation         | Name     | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |          |     2 |    52 |     2   (0)| 00:00:01 |
|*  1 |  TABLE ACCESS FULL| STUDENTS |     2 |    52 |     2   (0)| 00:00:01 |
-------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - filter("FACULTYNAME"='Elektronski' AND "FACULTYID"='1')
```

Nakon postavljanja OPTIMIZER_DYNAMIC_SAMPLING parametra na 6, rezultati su sledeći:

```
ALTER SESSION SET optimizer_dynamic_sampling=6;
```

```
Plan hash value: 4078133427

-------------------------------------------------------------------------------
| Id  | Operation          | Name     | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |          |     4 |  1600 |     2   (0)| 00:00:01 |
|*  1 |  TABLE ACCESS FULL| STUDENTS |     4 |  1600 |     2   (0)| 00:00:01 |
-------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - filter("FACULTYNAME"='Elektronski' AND "FACULTYID"='1')

Note
-----
   - dynamic statistics used: dynamic sampling (level=2)
```

Sa standardnom statistikom, optimizator procenjuje kardinalnost kao 2 reda, a zapravo postoje 4 reda koja zadovoljavaju uslov. Bez proširene statistike, optimizator ne zna da postoji veza između FacultyName i FacultyId (“Elektronski” ima id 1), ali postavljanjem optimizer_dinamic_sampling na nivo 6, optimizator koristi dinamičko uzorkovanje radi prikupljanja dodatnih informacija. Ove dodatne informacije omogućavaju optimizatoru da generiše tačniju procenu kardinalnosti.

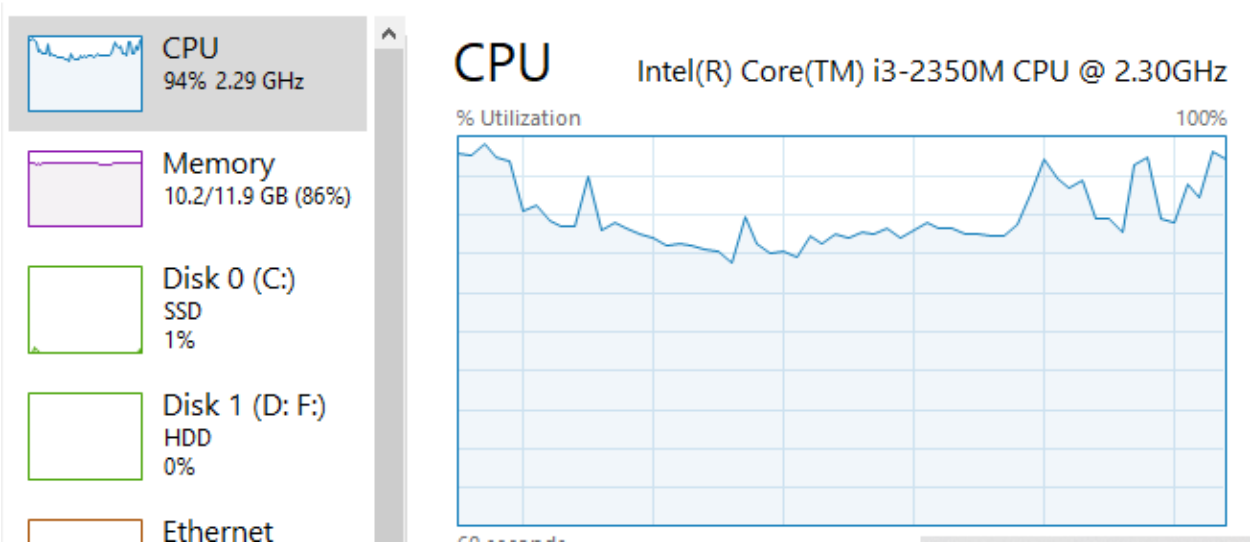
Statistika sistema

Sistemska statistika je uvedena kako bi se omogućilo optimizatoru da preciznije “naplati” svaku operaciju u planu izvršenja. Ovo je omogućeno time što optimizator ima pristup informacijama o stvarnom sistemskom hardveru koji izvršava upit. Informacije o sistemskom hardveru bi obuhvatale vrednosti kao što su brzina procesora i IO performanse (ulazno-izlazne performanse).

Sistemska statistika je podrazumevano omogućena i automatski se inicijalizuje podrazumevanim vrednostim (vrednostima koje su reprezentativne za veliki broj sistema). Kada se prikupe sistemske statistike, zameniće inicijalne vrednosti. Za prikupljanje sistemske statistike može se koristiti DBMS_STATS.GATHER_SYSTEM_STATS (preporuka je da se procedura

izvrši u trenutku velikog opterećenja sistema). Sistemsku statistiku treba prikupiti samo jednom. Sistemska statistika se ne prikuplja automatski kao deo posla automatskog prikupljanja statistike.

Kao što je već napomenuto, preporuka je skupiti statistiku sistema kada je on pod najvećim opterećenjem i to u vremenskom periodu od 1h.



Procedura kojom se pokreće skeniranje:

```
execute dbms_stats.gather_system_stats('Start');
```

Nakon sat vremena, treba se pozvati procedura koja će zaustaviti prikupljanje podataka.

```
execute dbms_stats.gather_system_stats('Stop');
```

Prikupljena statistika nalazi se u aux.stats$ tabeli.

	PNAME	PVAL1
1	STATUS	(null)
2	DSTART	(null)
3	DSTOP	(null)
4	FLAGS	1
5	CPUSPEEDNW	1095.53739786298
6	IOSEEKTIM	10
7	IOTFRSPEED	4096
8	SREADTIM	0.388
9	MREADTIM	(null)
10	CPUSPEED	1696
11	MBRC	(null)
12	MAXTHR	(null)
13	SLAVETHR	(null)

- CPUSPEEDNW - brzina procesora - MHz
- IOSEEKTIM -ulazno/izlazno vreme traženja u ms
- IOTFRSPEED - ulazno/izlazna brzina prenosa u ms

Statistika vezana za opterećenje:

- SREADTIM - vreme čitanja jednog bloka u ms
- MREADTIM - vreme čitanja više blokova u ms
- CPUSPEED - brzina procesora
- MBRC - prosečan broj pročitanih blokova kada se čita više blokova
- MAXTHR - maksimalni ulazno/izlazni protok

Online prikupljanje statistike

Od Oracle 12c verzije, statistike tabele se prikupljaju automatski za neke grupne operacije. Ova nova funkcija pod nazivom „Online Statistics Gathering for Bulk Loads“ može biti veoma praktična. Statistika tabele se automatski prikuplja za sledeće naredbe:

- CREATE TABLE AS SELECT
- Direct-Path INSERT naredba za praznu tabelu

Create table as select

Prikupljanje statistike uz pomoć metoda kreiranja tabele preko select naredbe može se videti na slikama dole. Na prvoj slici je prikazano konkretno kreiranje tabele na osnovu već postojeće.

```
CREATE TABLE online_persons AS
SELECT * FROM Persons;
```

Nakon kreiranja, pregledom sadržaja USER_TAB_STATISTICS tabele za upravo kreiranu tabelu, moguće je videti da je statistika već zapisana.

```
SELECT table_name, num_rows, last_analyzed
  FROM user_tab_statistics
 WHERE table_name = 'ONLINE_PERSONS';
```

	TABLE_NAME	NUM_ROWS	LAST_ANALYZED
1	ONLINE_PERSONS	7	13-APR-22

Direct-Path INSERT

Direct-Path INSERT je ovde iskorišćen kao INSERT sa dodavanjem.

```
INSERT INTO online_persons
SELECT * FROM Persons;
```

Slika ispod pokazuje da je nakon ovog dodavanja statistika ostala nepromenjena. Ovo se dešava jer tabela nad kojom se izvršila naredba prethodno nije bila prazna. Odnosno, ponovno prikupljanje statistike nakon INSERT naredbe se vrši samo ukoliko je tabela prethodno bila prazna ili je nad tabelom izvršena TRUNCATE naredba.

	TABLE_NAME	NUM_ROWS	LAST_ANALYZED
1	ONLINE_PERSONS	7	13-APR-22

Realtime prikupljanje statistike

Oracle 19c verzija uvodi statistiku u realnom vremenu, koja proširuje prikupljanje online statistike tako da uključuje i konvencionalne DML naredbe. Statistike se obično prikupljaju automatskim u okviru održavanja baze podataka(samo jednom dnevno).

Pošto statistika može da zastari između DBMS_STATS poziva, statistika u realnom vremenu pomaže optimizatoru da generiše optimalnije planove.

Reference

https://www.dbta.com/Columns/DBA-Corner/Database-Statistics-and-Optimization-54605.aspx
https://docs.oracle.com/cd/A97630_01/server.920/a96533/stats.htm
https://docs.oracle.com/cd/A84870_01/doc/server.816/a76992/stats.htm
https://tutorialitpoint.blogspot.com/p/dbmsstats-dbmsstats-package-dbmsstats.html
http://www.dba-oracle.com/t_gather_stats_job.htm
https://smarttechways.com/2014/09/18/restore-old-statistics-of-the-table-in-oracle/
https://blogs.oracle.com/optimizer/post/dynamic-sampling-and-its-impact-on-the-optimizer
http://www.dba-oracle.com/t_dbms_stats_compare_statistics.htm
https://uhesse.com/2012/04/23/diff_table_stats_in_history-example/
http://www.dba-oracle.com/t_dbms_stats_gather_system_stats.htm
https://danischnider.wordpress.com/2015/12/23/online-statistics-gathering-in-oracle-12c/