Knjiga

1

FREE ANONIMA – SAMOSTALNA OBUKA ORACLE

http://stroga.anonima.free.fr

Oracle8i DBA Arhitektura i administriranje

ORACLE8I DBA (1Z0-023)

Arhitektura i administriranje Oracle8i servera baze podataka

Sadrzaj

Poglavlje

1

Arhitekturalne komponente Oracle servera baze podataka

U ovome poglavlju su prikazane osnovne arhitekturalne komponente jednog ORACLE servera baze podataka. Kao buduci administratori baze podataka trebacete vec u samom startu da ovladate ovim osnovnim pojmovima te da se upoznate sa zadacima koje cete obavljati u toj funkciji.Potpuno razumijevanje ovih osnovnih komponenti servera Oracle i njihovih medjusobnih relacija je neophodno za dalji rad.

Uvod u Oracle Server

Oracle Server je baza podataka koju odlikuje visoka raspolozivost i efikasnost pri radu sa znacajnim brojem korisnika i velikim obimom informacija. Oracle Server je kljucni element u izradi informacionih sitema koji sadrzi sve podatke neophodne za pravovremeno i efikasno djelovanje preduzeca. Ovakvo centralno mjesto baze podataka u jednom informacionom sistemu podrazumijeva njeno efikasno i redovno odrzavanje. Kao sto cete vidjeti u daljem tekstu, osiguravanje raspolozivosti, korektnog i efikasnog funkcionisanja baze podataka su zadatak administratora.

Zadaci administratora baze podataka

U Oracle okruzenju administrator baze podataka je osoba koja je zaduzena za upravljanje sistemom i njegovim korisnicima. Administrator baze podataka obavlja brojne zadatke prilikom upravljanja bazom podataka i njenim korisnicima.

- Jedan od najvaznijih zadataka administratora baze podataka je instaliranje Oracle servera i aplikativnih alata. Nakon instaliranja Oracle servera, administrator baze podataka je takodjer duzan planirati i kreirati bazu podataka.

- Administrator baze podataka je duzan osigurati raspolozivost baze podataka za sve njene korisnike.

- Nakon sto programeri kreiraju aplikaciju, zadatak administratora baze podataka je da kreira elemente logicke strukture skladistenja podataka kako sto su tablespace-i, table, pogledi i indeksi koje zahtijeva pomenuta aplikacija.

- Pored ovoga, administrator baze podataka upravlja i elementima fizicke strukture skladistenja kao sto su datoteke podataka, kontrolne datoteke i redo-log datoteke.

- Zadatak administratora baze podataka je da namijeni potreban prostor za skladistenje sistema (system storage) i da planira buduce prostorne zahtijeve sistema baze podataka.

- Upravljanje skladisnim prostorom zasnovanom na projektnim specifikacijama je takodjer jedan od zadataka koje obavlja administrator baze podataka. Oslanjajuci se na informacije koje dobija od kreatora aplikacija, administrator baze podataka ce takodjer izmjeniti strukturu baze podataka kada je to neophodno.

- Oracle baza podataka ima veci broj korisnika. Administrator baze podataka je zaduzen za upravljanje, nadziranje i kontrolu rada koji obavljaju ovi korisnici.

- Osim sto upravlja postojecim, administrator baze podataka moze da upise nove korisnike baze podataka.

- U odredjenim situacijama, osiguranje sigurnosti sistema moze biti zadatak administratora baze podataka. U velikim sistemima uobicajeno je da ovu ulogu obavlja zasebna osoba, a ne administrator baze podataka.

- Buduci da bazu podataka istovremeno koristi veci broj osoba, u cilju osiguranja efikasnog funkcionisanja baze podataka administrator baze podataka je duzan da nadzire i optimizira performance baze podataka.

- Jedna Oracle baza podataka sadrzi velike kolicine podataka. U slucaju kvara sistema moze doci do gubljenja strateskih i nuznih informacija. Da bi se sprijecilo ovo gubljenje informacija, zadatak administratora je da planira i izvrsava sigurnosne kopije (backups) baze podataka. U slucaju da dodje do pada sistema, zadatak administratora baze podataka je da obavi brzi oporavak (recovery) baze podataka.

Prikljucivanje na Oracle Server

Korisnik koji zeli da pribavi informacije iz baze podataka mora prije svega da se prikljuci na Oracle server. Postoje tri nacina na koje se korisnik baze podataka moze prikljuciti na Oracle server : lokalni, dvoslojni i troslojni.

Koristeci lokalni prikljucak, korisnik se moze direktno prijaviti na masinu na kojoj je instaliran Oracle server. Npr. korisnik se prikljucuje na UNIX masinu na kojoj je instaliran Oracle server i pomocu alata Server Manager pristupa bazi podataka.

Koristeci dvoslojni prikljucak, poznat jos kao i klijent/server prikljucak, korisnik se prijavljuje na masini koja je direktno vezana sa masinom na kojoj je instaliran Oracle server. Primjer dvoslojnog prikljucka je kada korisnik koji iz aplikacije Oracle Forms na masini Windows 9x pristupa Oracle bazi koja se nalazi na NT serveru.

U troslojnom prikljucku masina korisnika baze podataka je prikljucena na mrezni (aplikativni) server. Ovaj mrezni server je putem mreze prikljucen na masinu na kojoj je instaliran Oracle server. Primjer troslojnog prikljucka je kada korisnik baze podataka pomocu internet navigatora na svojoj masini startuje aplikaciju na NT serveru. Server NT potom pribavlja podatke iz baze na UNIX serveru.

Kada se korisnik prikljucuje na masinu na kojoj funkcionise Oracle server, dva procesa su ukljucena, jedan korisnicki i jedan server proces.

Korisnicki proces je kreiran kada korisnik pokrene alat kao sto je SQL*Plus ili aplikaciju izradjenu pomocu alata kao sto je npr. Oracle Forms. Korisnicki proces je mehanizam koji sluzi za izvrsavanje koda aplikativnog programa kao sto je Oracle Forms aplikacija ili nekog Oracle alata kao sto je OEM. U klijent/server modelu Oracle alat ili aplikacija se izvrsavaju na klijent masini.

Server proces je kreiran kada se korisnik prijavi na Oracle server precizirajuci svoje ime korisnika, lozinku i ime baze podataka. Server proces je kreiran na istoj masini na kojoj funkcionise Oracle server. Server proces je mehanizam interakcije izmedju Oracle servera i korisnickog procesa na masini klijentu. Server proces obavlja na Oracle serveru operacije koje zahtijeva korisnicki proces.

Komunikacijski put izmedju korisnickog procesa i Oracle servera nazivamo prikljuckom. Kada korisnik i Oracle server djeluju medjusobno prikljucak je uspostavljen.

Ukoliko korisnik koristi alat ili aplikaciju na istoj masini na kojoj se nalazi i Oracle server tada se komunikacijski put uspostavlja koristeci komunikacijski mehanizam medju procesima raspoloziv na Oracle serveru - Inter Process Communication (IPC).

Ako korisnik koristi alat ili aplikaciju na klijent masini, za komunikaciju izmedju korisnika i Oracle servera putem mreze neophodna je upotreba mreznog softvera.

Zaseban prikljucak korisnika na Oracle server nazivamo sesijom. Sesija pocinje od momenta kada Oracle server potvrdi identitet korisnika a zavrsava se odjavljivanjem korisnika ili nenormalno.

Vise istovremenih sesija moze postojati za korisnika koji se prikljuci na Oracle server sa razlicitih terminala.

Korisnicki procesi

Da bi izvrsio aplikativni alat kao sto je SQL*Plus ili aplikaciju uradjenu pomocu alata kao sto je Oracle Forms, Oracle kreira korisnicki proces. Korisnicki proces se takodjer naziva i klijentom. Ovaj proces je izvrsavan na masini na kojoj se korisnik direktno prijavljuje.

Korisnicki proces je pokrenut kada korisnik startuje alat a zavrsava se odjavljivanjem korisnika ili uslijed nenormalnog zavrsetka rada kao sto je pad sistema.

Korisnicki proces sadrzava User Program Interface (UPI). UPI je mehanizam koji korisnicki proces koristi za komuniciranje sa server procesom.

UPI je takodjer standardna metoda komuniciranja ma kojeg klijent alata ili aplikacije sa Oracle softverom.

UPI generise pozive Oracle serveru svaki put kada korisnik nacini neki zahtjev. Nakon sto je zahtjev obradjen na Oracle serveru, rezultat je uzvracen korisnickom procesu.

Server procesi

Kao sto smo vidjeli, korisnicki proces proslijedjuje zahtjeve korisnika server procesu. Server proces manipulira ovim korisnickim zahtjevima. Server proces je pokrenut na istoj masini na kojoj funkcionise Oracle server.

U Oracle serverima posebne namjene (*dedicated*), server proces opsluzuje samo jedan korisnicki proces. Server proces je kreiran kada korisnik zahtijeva prikljucak, a zavrsava se kada se korisnik odjavi.

Svaki server proces koristi dio memorije koji nazivamo Program Global Area (PGA). PGA je kreirana kada je pokrenut server proces.

Server proces ukljucuje Oracle Program Interface (OPI) koji je koristen za komuniciranje sa Oracle serverom na zahtijev korisnickog procesa.

Server proces uzvraca korisnickom procesu informacije o statusu i rezultate zahtjeva.

Klijent/server sistemi razdvajaju korisnicke i server procese i izvrsavaju ih na odvojenim masinama.

U multithread konfiguraciji servera (MTS – multithreaded server) vise korisnickih procesa mogu zajednicki koristiti jedan server proces, sto nije slucaj u dedicated konfiguraciji servera.

Oracle instanca i baza podataka

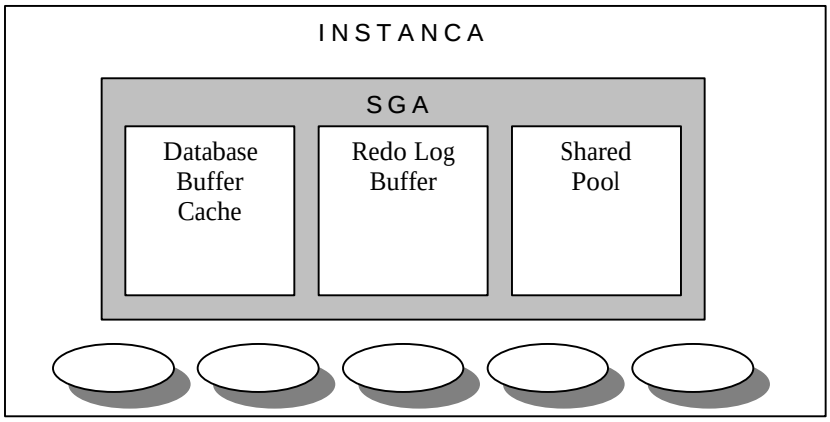
Globalno uzevsi Oracle Server se sastoji iz dvije glavne komponente : baze podataka i instance. Nerijetko dolazi do konfuzije prilikom upotrebe ova dva pojma gdje se na mjesto jednog koristi drugi i obratno. Jedan od razloga za ovu konfuziju je i taj da, u vecini slucajeva, postoji relacija jedan-na-jedan izmedju ova dva pojma. Ovo znaci da uvijek, osim u slucaju Oracle Parallel Servera, jedna instanca pristupa jednoj bazi podataka. U slucaju izuzetka koji je Oracle Parallel Server, vise instanci pristupaju jednoj bazi podataka.

U daljem tekstu cemo se detaljnije upoznati sa Oracle bazom podataka. i sa instancom te ponaosob sa svim elementima koji sacinjavaju ove dvije osnovne komponente Oracle Servera.

Sistem Global Area (SGA)

Oracle server se sastoji iz Oracle instance i Oracle baze podataka. Oracle instanca sadrzi memorijsku zonu koju nazivamo *System Global Area (SGA).* SGA sadrzi podatke i kontrolne informacije Oracle servera. SGA nazivamo takodjer i *Shared Global Area.*

Oracle rezervise memoriju za SGA kada je instanca startovana i oslobadja memoriju kada je instanca zaustavljena. Memorijsko podrucje koje je rezervisano za SGA koriste zajednicki svi korisnici prikljuceni na Oracle bazu podataka i ona se treba uklopiti u stvarnu memoriju koja nije ni swap-ovana ni pagirana za potrebe efikasnog funkcionisanja. Svaka instanca posjeduje svoju vlastitu SGA.



SLIKA 1.1 shematski prikaz strukture jedne Oracle instance

SGA sadrzi nekoliko grupa memorijskih struktura. Ove grupe su kreirane prilikom podizanja instance. Tri glavne memorijske strukture su : Shared Pool, Database Buffer Cache i Redo Log Buffer.

Shared Pool sluzi za pohranjivanje informacija kao sto su najskorije izvrsene SQL instrukcije i skoro koristene informacije rjecnika podataka.

Database Buffer Cache poznat kao Data Buffer Cache sluzi za pohranjivanje najskorije koristenih podataka korisnika. On sadrzi u memoriji kopije blokova podataka radi postizanja efikasnog funkcionisanja baze.

Redo Log Buffer poznat i kao Log Buffer Cache sluzi za pohranjivanje promjena izvrsenih u bazi podataka.

Memorijska zona SGA je read-write, sto znaci da je moguce iz nje citati i u nju upisivati podatke. Bilo ko prikljucen na bazu podataka moze citati u SGA. dok jedino skup pozadinskih procesa moze u njoj pisati. U ovaj skup procesa ulaze server procesi te SMON, PMON i RECO.

Pozadinski procesi

Jedna Oracle instanca sastoji se od SGA i skupa pozadinskih procesa (na slici 1.1 pozadinski procesi predstavljeni su elipsama). Svaki put kada je instanca Oracle podignuta, Oracle za nju kreira skup pozadinskih procesa. Pozadinski procesi u jednoj instanci obavljaju rutinske funkcije neophodne za opsluzivanje zahtijeva koji dolaze istovremeno od strane vise korisnika.

Pozadinski procesi su procesi operativnog sistema u vecini operativnih sistema osim u Windows NT gdje se radi o thread-ima jednog procesa. Svaka instanca moze koristiti vise pozadinskih procesa ciji broj ovisi o konfiguraciji Oracle servera. U Oracle 8i postoji pet obaveznih pozadinskih procesa za svaku instancu.

Obavezni pozadinski procesi Oracle 8i su : Database Writer (DBWn), Log Writer (LGWR), System Monitor (SMON), Process Monitor (PMON) i Checkpoint (CKPT).

Database Writer (DBWn) proces upisuje izmjenjene podatke iz Database Buffer Cache-a u datoteke podataka.

Log Writer (LGWR) proces upisuje registrovane izmjene iz redo-log buffer-a u redo-log datoteke. Te izmjene su nam poznate pod nazivom redo-log podaci.

System Monitor (SMON) proces obavlja oporavak (*recovery*) instance prilikom njezina podizanja. U sistemu sa vise instanci, SMON proces jedne instance obavlja oporavak drugih instanci koje su pretrpile pad. Pored ovoga, zadatak SMON procesa je i da obavi ciscenje privremenih segmenata koji vise nisu u upotrebi te da oporavi « mrtve » transakcije ispustene prilikom pada sistema do kojeg je doslo uslijed neke greske. SMON takodjer ujedinjuje fragmentirani slobodni prostor u datotekama podataka.

Process Monitor (PMON) proces obavlja oporavak (*recovery*) procesa kada neki od korisnickih procesa pretrpi neuspjeh. PMON proces je odgovoran za ciscenje cache memorije i oslobadjanje resursa sistema koje je drzao korisnicki proces.

Checkpoint (CKPT) proces je odgovoran za azuriranje informacije o statusu baze podataka, kao sto su zaglavlja datoteka podataka. Ovo se desava svaki put kada su izmjenjeni podaci iz Database Buffer Cache-a trajno upisani u datoteke podataka

tokom checkpoint dogadjaja ili nakon log switch-a (izmjene aktivne redo-log datoteke).

Pored navedenih obaveznih procesa postoje i neobavezni pozadinski procesi kao sto su : Recoverer (RECO), Archiver (ARCH), Lock (LCKn) i Dispatcher (Dnnn).

Recoverer (RECO) proces je koristen za rijesavanje distribuiranih transakcija koje su nerijesene uslijed mreznog ili sistemskog problema na distribuiranoj bazi podataka.

Archiver (ARCH) proces kopira popunjene online redo-log datoteke u arhivu. ARCH proces je aktivan jedino kada je baza podataka ARCHIVELOG nacinu rada.

Lock (LCKn) proces sluzi za rezervisanje izmedju instanci u Oracle Parallel Serveru u kojem istodobno funkcionise vise instanci. Mozemo konfigurisati ma koji lock proces izmedju LCK1 i LCK9.

Dispatcher (Dnnn) je neobavezan proces koji je koristan jedino u multithreaded (MTS) konfiguraciji Oracle servera. Dnnn proces dopusta korisnickim procesima da zajednicki koriste server proces u MTS konfiguraciji. Moze da postoji ma koji broj dispacher procesa izmedju D000 i Dnnn.

Fizicka struktura (datoteke) baze podataka

Kao sto je vec receno Oracle Server posjeduje dvije osnovne komponente: bazu podataka i instancu. Jako cesto su ova dva pojma zamjenjivana ili pogresno upotrebljavana. Donekle je ova zbrka uzrokovana odnosom jedan-na-jedan koji postoji u vecini slucajeva izmedju jedne Oracle baze podataka i Oracle instance. Drugim rijecima, sa izuzecem Oracle Parallel Servera gdje vise Oracle instanci pristupaju jednoj Oracle bazi podataka, uvijek jedna Oracle instanca odgovara jednoj Oracle bazi podataka.

Oracle baza podataka je kolekcija podataka koja sluzi da pohranjujemo i ponovno pronalazimo medjusobno povezane informacije. Ona se sastoji od fizickih struktura skladistenja, odnosno datoteka. Jedan od zadataka administratora baze podataka je rukovanje ovim datotekama.

Da bismo bili u stanju raspolagati ovim datotekama moramo prije svega da se upoznamo sa postojecim tipovima ovih datoteka i podacima koji su u njima pohranjeni. Postoje tri tipa ovih datoteka : datoteke podataka, kontrolne datoteke i redo-log datoteke.

Datoteke podataka

Oracle baza podataka posjeduje barem jednu *datoteku podataka* koja odgovara SYSTEM tablespaceu . U datotekama podatka su pohranjeni sistemski podaci poznati i pod imenom *rjecnik podataka* te objekti korisnika baze podataka i originalne vrijednosti *(before images*) blokova podataka koji su izmjenjeni u tekucim transakcijama.

Podaci iz datoteka podataka su ucitavani, svaki put kada je to zahtijevano i spremljeni u u memorijsku strukturu nazvanu Database Buffer Cache. Izmjenjeni i novi podaci nisu trenutacno upisivani u datoteke podataka. U cilju efikasnijeg funkcionisanja baze podataka oni su spremani Database Buffer Cache. Baza podataka, odnosno DBWn proces koji je zaduzen za upisivanje ovih podataka u datoteke podataka praktikuje upisivanje sa odgodom (*deferred write model*) . Na ovaj nacin smanjuje se broj I/O pristupa disku i poboljsava efikasnost rada baze podataka.

Redo-log datoteke

Redo-log datoteke su koristene za biljezenje promjena izvrsenih nad podacima (instrukcijama insert, update ili delete) i osiguravanje oporavka podataka u slucaju pada instance. Svaka redo stavka (*redo entry*) sadrzi informaciju nazvanu vektorom promjene (*change vector*) koja odgovara jednoj izmjeni podataka.

Redo-log datoteke su prvenstveno koristene za oporavljanje baze podataka nakon kvara instance ili diska. Obzirom da su sve izmjene zabiljezene u redo-log datotekama, svaki kvar je moguce opraviti koristeci se ovim informacijama.

Svaka Oracle baza podataka posjeduje najmanje dvije redo-log grupe od kojih svaka posjeduje barem jednu redo-log datoteku. Da bi se zastitili od gubljenja redo-log datoteka uslijed kvarova na disku Oracle nam omogucuje umnozavanje (multipleksiranje) redo-log datoteka. U ovom slucaju moguce je drzati dvije ili vise redo-log datoteka na razlicitim diskovima. Kopije redo-log datoteka drzane na razlicitim diskovima nazivamo *mirror log* datotekama. Grupa multipleksiranih redo-log datoteka naziva se redo-log grupom. Svaki clan u redo-log grupi ima istu velicinu i informacije su istovremeno u njih upisivane.

Interes drzanja redo-log datoteka na razlicitim diskovima lezi u tome da u slucaju kvara na jednom disku mozemo i dalje raspolagati datotekama na preostalim diskovima.

Kontrolne datoteke

Kontrolne datoteke sadrze informacije neophodne za odrzavanje i provjeru integriteta baze podataka. One sadrze strukturu baze podataka, kao sto su ime baze podataka, vremenski zig (*timestamp*) kreiranja baze podataka, imena i lokacije datoteka podatka i redo-log datoteka kao i njihovo stanje (online ili offline). Kontrolne datoteke sadrze takodjer i log broj trenutacno koristene redo-log datoteke.

Informacije pohranjene u kontrolnim datotekama su koristene za identifikovanje datoteka podataka i redo-log datoteka koje moraju biti otvorene kada otvaramo bazu podataka. Ukoliko je struktura baze podataka izmjenjena Oracle automatski mijenja kontrolne datoteke kako bi one odrazile ove izmjene.

Kontrolne datoteke su koristene prilikom pokretanja baze podataka kako bi se provjerilo da li su sve datoteke koje sacinjavaju bazu podataka raspolozive. Obzirom da je aktualni log broj prisutan i u zaglavlju svih datoteka podataka,

kontrolna datoteka moze biti koristena i za provjeru konzistentnosti datoteka podataka prilikom pokretanja baze podataka.

Kontrolne datoteke su koristene i prilikom oporavka (*recovery*) baze podataka ukoliko na njoj postoji nedostatak. Svaka Oracle baza podataka zahtijeva barem jednu kontrolnu datoteku. Medjutim, i ovdje kopije kontrolne datoteke mogu biti drzane na razlicitim diskovima.

Velicina kontrolne datoteke je promjenjiva i ovisi o definisanim vrijednostima pojedinih ogranicenja baze podataka kao i od toga da li Recovery Manager koristi kontrolnu datoteku kao referencijal (*repository*).

Napomena : Datoteke podataka, redo-log datoteke i kontrolne su jedine datoteke koje sacinjavaju Oracle bazu podataka. Iako Oracle koristi i druge vazne datoteke, kao sto su datoteke parametara ili Net8 konfiguracijske datoteke, one nisu smatrane komponentama baze podataka.

Ostale datoteke

Oracle server koristi i druge datoteke koje su neophodne za operacije kao sto su podizanje instance ili identifikovanje korisnika. Ove datoteke su drugi vid strukture fizickog pohranjivanja koje koristi Oracle server. Postoje tri tipa ovih datoteka : datoteka parametara, datoteka lozinki i arhivirane redo-log datoteke. Oracle takodjer generise ALERT i TRACE datoteke. To su datoteke u koje Oracle upisuje izvjestaje o statusu i greskama. ALERT datoteke pruzaju vazne informacije o statusu instance i proizvedenim greskama.

Datoteka parametara

Datoteka parametara sluzi za definisanje karakteristika Oracle instance. Prilikom pokretanja instance Oracle server se sluzi datotekom parametara kako bi u njoj pribavio konfiguracijske informacije instance. Konfiguracija instance je definisana vrijednostima inicijalizacijskih parametara u datoteci parametara.

Inicijalizacijski parametri definisu ime baze podataka, kolicinu memorije koju treba rezervisati za instancu, imena kontrolnih datoteka i brojne druge parametre.

Datoteka lozinki

Datoteka lozinki je koristena za kontrolu identiteta privilegovanih korisnika Oracle baze podataka.

Arhivirane redo-log datoteke

Arhivirane redo-log datoteke sluze za pohranjivanje *offline* kopija redo-log datoteka koje koristimo prilikom oporavka medija.

Napomena: Proces primjene redo-log zapisa prilikom operacije oporavka naziva se *rolling forward*. Nakon sto su primjenjene redo-log datoteke, rollback segmenti su koristeni za identifikovanje i anuliranje nepotvrdjenih transakcija ubiljezenih u redo-log. Ovaj proces anuliranja nepotvrdjenih transakcija naziva se *roll back*.

Oracle kreira arhivirane redo-log datoteke tako sto arhivira redo-log datoteke jednom kada su popunjene. Nakon sto su arhivirane, online redo-log datoteke mogu biti nanovo koristene.

Ako baza podataka funkcionise u ARCHIVELOG nacinu rada, redo-log datoteke su arhivirane prije nego sto su nanovo koristene u ciklicnom slijedu. Ako, naprotiv, baza podataka funkcionise u NOARCHIVELOG nacinu rada, redo-log datoteke nisu arhivirane prije nego sto su ponovo koristene (njihov sadrzaj je pogazen novim upisima).

U ARCHIVELOG nacinu rada baza podataka moze biti potpuno oporavljena i nakon kvara instance i nakon kvara medija.

Obrada upita u bazi podataka

Sigurno ste vec imali priliku koristiti alat kao sto je na primjer SQL*Plus u cilju pribavljanja informacija iz Oracle baze podataka. Pomenuta aplikacija vam pruza mogucnost da jednostavno odasiljete zahtijeve Oracle Serveru koristeci se pri tome SQL instrukcijama koje pisete u editoru teksta aplikacije i potom ih izvrsavate jednostavnim pritiskom na tipku Enter. Medjutim, od momenta kada ste potvrdili vas zahtijev tipkom Enter do trenutka kada vam Oracle Server uzvrati rezultat koji odgovara vasem zahtijevu desava se iza kulisa citav niz operacija. Jedan bezazleni zahtijev tipa « zelim znati imena radnika zaposlenih u toku ove godine » ce uzrokovati niz interakcija razlicitih arhitekturalnih komponenti Oracle baze podataka kao sto je detaljno predstavljeno u daljem tekstu.

Etape obrade upita u bazi podataka

Kada korisnik postavi jednu SQL instrukciju ili upit (query), korisnicki proces proslijedjuje te instrukcije ili upite server procesu. Server proces prima instrukcije i upite i pristupa njihovoj obradi.

Postoje tri glavne etape u obradi jednog upita : analiziranje (*parse*), izvrsenje (*execute*) i dobavljanje (*fetch*).

Analiziranje upita

U ovoj etapi server proces najprije prima upit od korisnickog procesa.

Potom, server proces provjerava sintaksu upita kako bi se uvjerio u njegovu valjanost.

Server proces provjerava takodjer i sigurnosne privilegije korisnika u pogledu pristupa objektima na koje se poziva u upitu. Server proces koristi Shared Pool memorijsku strukturu iz SGA kako bi u njoj kompilirao instrukciju i izgradio kompiliranu verziju instrukcije (*parse tree*).

Na kraju ove etape server proces uzvraca korisnickom procesu informaciju o statusu. Ova informacija ukazuje na to da li je analiziranje upita bilo uspjesno ili ne.

Izvrsenje upita

U ovoj fazi server proces priprema dobavljanje zahtjevanih podataka.

Ukoliko je obradjivana instrukcija DELETE ili UPDATE server proces ce zabraniti pristup (*lock*) svim linijama kojih se tice ovaj zahtjev, tako da ostali korisnici baze podataka ne mogu da im pristupe.

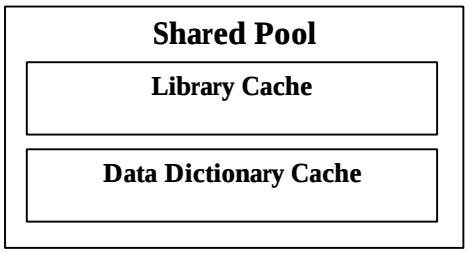
Dobavljanje

U ovoj fazi server povlaci linije koji se zahjev tice i proslijedjuje ih korisniku. Ovisno o kolicini memorije potrebne za prenos rezultata upita, dobavljanje se ce obaviti iz jednog ili vise puta.

Shared Pool

Etapa analiziranja upita sluzi se Shared Pool-om, memorijskom strukturom koja je dio SGA. Shared Pool sadrzi komponente zajednicke memorije koje server proces koristi za analiziranje SQL instrukcija.

Komponente zajednicke memorije u Shared Pool-u su Library Cache i Data Dictionary Cache.



SLIKA 1.2 prikaz komponenti Shared Pool memorijske zone

Library Cache sadrzi informacije u vezi najskorije koristenim SQL instrukcijama. U njoj su pohranjeni tekst instrukcije, kompilirana verzija instrukcije (parse tree) te plan izvrsenja instrukcije (*execution plan*). Plan izvrsenja komande sadrzi korake izvrsenja instrukcije koje odredjuje optimizator.

U slucaju da je SQL instrukcija, za koju postoje informacije (kompilirana verzija i plan izvrsenja) u Library Cache-u, nanovo odaslana od strane ma kojeg korisnickog

procesa prije nego sto njen plan izvrsenja zastari, njena kompilirana verzija i plan izvrsenja mogu biti ponovo koristeni.

Library Cache sadrzi i druge kontrolne strukture kao sto su zabrane pristupa (*locks*) i Library Cache Handles.

Data Dictionary Cache sadrzi informacije rjecnika podataka u vezi objekata i struktura baze podataka te definicija kolona.

Data Dictionary Cache sadrzi takodjer i vrijedeca imena korisnika, njihove lozinke i privilegije za bazu podataka.

Data Dictionary Cache je poznat i pod imenom Row Cache zato sto su u njemu podaci sadrzani u linijama a ne u medjuspremnicima (*buffer*) koji sadrze cijele blokove podataka.

Database Buffer Cache

Prilikom obrade upita, server proces provjerava prisustvo blokova podataka neophodnih za obradu upita u Database Buffer Cache-u, koji je memorijska struktura i dio SGA.

Database Buffer Cache sadrzava blokove podataka koji su ucitani iz datoteka podataka. Ukoliko server proces ne uspije pronaci zahtijevani blok podataka u Database Buffer Cache-u, on ce taj blok podataka ucitati iz datoteke podataka i njegovu kopiju spremiti u Database Buffer Cache.

Sa stanovista efikasnosti rada baze podataka, pozeljno je da server proces pronalazi potrebne podatke u memoriji (cache hit). Ukoliko to nije slucaj (cache miss) server proces ce pristupati podacima citajuci ih sa diska sto je znatno sporije u odnosu na pristup podacima u memoriji.

Database Buffer Cache nazivamo jos i Buffer Cache. On je koristen za pohranjivanje najskorije i cesto koristene blokove podataka.

Database Buffer Cache je skup medjuspremnika (*buffer*-a) baze podataka. Velicina svakog od medjuspremnika u Database Buffer Cache-u je jednaka velicini jednog bloka podataka u bazi podataka.

Velicina memorijske zone Database Buffer Cache jednaka je proizvodu broja medjuspremnika (*buffer*-a) u Database Buffer Cache-u i velicine bloka podataka u bazi podataka.

Velicina Database Buffer Cache-a = DB_BLOCK_BUFFERS * DB_BLOCK_SIZE

Broj medjuspremnika (*buffer*-a) u Database Buffer cache-u odredjen je vrijednoscu parametra DB_BLOCK_BUFFERS. S obzirom da je velicina DB_BLOCK_SIZE odredjena prilikom kreiranja baze podataka, te da je DB_BLOCK_BUFFERS dinamicki parametar, da bismo izmjenili velicinu Database Buffer-a neophodno je zaustaviti i nanovo podici instancu.

Database Buffer Cache sadrzi kao izmjenjene tako i neizmjenjene blokove podataka.

Da bi napravio mjesta za nove blokove podataka u database Buffer Cache-u, Oracle koristi algoritam Last Recently Used (LRU) da bi oslobodio prostor koji zauzimaju medjuspremnici sa podacima koji nisu skoro bili koristeni.

Database Buffer je podjeljen u dvije liste: necistu (*dirty*) listu i najskorije koristenu (*last recently used*) LRU listu.

Necista lista sadrzi kopije blokova koji su izmjenjeni u odnosu na njihov sadrzaj na disku i koji jos uvijek nisu upisani nazad u datoteku podataka. DBWn proces primjenjuje odgodjeno upisivanje, kako je to vec napomenuto ranije, radi efikasnijeg rada baze podataka.

LRU lista sadrzi medjuspremnike koji mogu da su ili trenutacno koristeni ili slobodni za upotrebu ali isto tako i modifikovani medjuspremnici koji jos nisu prebaceni u necistu listu. Do njihovog premjestanja u necistu listu ce doci kada neki drugi proces bude trazio slobodne medjuspremnike u LRU listi.

Kada jedan proces nastoji da rezervise medjuspremnike u Database Buffer-u, on ce traziti slobodne medjuspremnike u LRU listi. Taj proces ce skanirati LRU listu dok ne pronadje slobodne medjuspremnike ili dok ne bude dostgnut odredjeni prag. Prilikom ovog skaniranja svi pronadjeni izmjenjeni medjuspremnici ce biti prebaceni u necistu listu. Ukoliko slobodni medjuspremnici nisu pronadjeni proces ce to signalirati DBWn procesu kako bi on reagovao tako sto ce upisati izmjenjene medjuspremnike na disk i na taj nacin osloboditi prostor u Database Buffer-u.

Database Buffer cache sluzi za minimiziranje I/O pristupa na disku i poboljsanje efikasnosti rada baze podataka.

U Oracle8i bazi raspolazemo sa vise zona u Database Buffer Cache-u. Objekte mozemo ucitati u jednu od slijedece tri zone:

- DEFAULT zona je koristena za objekte koji nisu pridruzeni odredjenoj zoni Database Buffer-a ili su, sto je prilicno rijetko, ekplicitno pridruzeni DEFAULT zoni. Velicina ove zone nije eksplicitno odredjenavec je za nju rezervisan prostor koji preostane u Database Buffer-u nakon rezervisanja prostora za preostale dvije zone: KEEP i RECYCLE.

- KEEP zona je koristena za objekte koje zelimo zadrzati u memoriji. Njena velicina je kontrolisana BUFFER_POOL_KEEP parametrom.

- RECYCLE zona je koristena za objekte koje zelimo zadrzati u memoriji tek toliko koliko je potrebno (dok ih algoritam ne proglasi zastarijelim). Njena velicina je kontrolisana BUFFER_POOL_RECYCLE parametrom.

Program Global Area (PGA)

Kada je startovan server proces, Oracle rezervise Program Global Area (PGA). PGA je memorijski medjuspremnik (*buffer*). On sadrzi podatke i kontrolne informacije za server proces ili za pojedinacni pozadinski proces.

PGA je nazivan i Process Global Area. PGA je rezervisan kada startuje server proces i oslobodjen kada se server proces zavrsi.

Jedna od karakteristika PGA jeste da je to memorijski region u koji je moguce pisati (*writeable*) i da je nedjeljiv (*non-sharable*).

Sadrzaj PGA ovisi o konfiguraciji Oracle servera. U konfiguraciji servera posebne namjene (*dedicated server*) PGA posjeduje zonu za razvrstavanje (*sort area*), informacije o sesiji, stanje kursora i stack prostor.

U multithreaded (MTS) konfiguraciji, pojedine informacije PGA su spremljene u SGA.

Zona za razvrstavanje, kao sto joj ime kaze, koristena je za obavljanje operacija razvrstavanje (*sort*), koje mogu biti neophodne prije obrade linija ili prije nego sto su linije odaslane korisnickom procesu.
Informacije o sesiji preciziraju privilegije korisnika za tekucu sesiju.

Stanje kursora ukazuje na fazu u kojoj se nalazi obrada pojedinih kursora koji su trenutacno koristeni u sesiji.

Kursor je poanter ka memorijskoj zoni pridruzenoj specificnoj SQL instrukciji. Oracle koristi radne zone za izvrsavanje SQL instrukcija i spremanje informacija o obradi. Kursori nam omogucuju da imenujemo ove radne zone i da pristupamo informacijama koje one sadrze.

Stack Space sadrzi varijable i matrice (tablice) sesije.

Obrada DML instrukcija

Pored instrukcije SELECT kojom pribavljamo informacije iz Oracle baze podataka na raspolaganju su nam i instrukcije koje nam omogucuju modifikovanje sadrzaja tih informacija. Pod modifikovanjem podrazumijevamo kako izmjenu sadrzaja informacija (azurirarnje) tako i dodavanje novih informacija te brisanje odnosno ponistavanje postojecih informacija. Tretman ovih instrukcija koje jos nazivamo i DML (Data Manipulation Language) instrukcijama razlikuje se od prethodno prikazanog tretmana instrukcija SELECT i ukljucuje i potrebu validiranja ili ponistavanja sacinjenih izmjena.

Izvrsna faza DML instrukcije

Oracle raspolaze instrukcijama za manipulisanje podataka kao sto su INSERT, UPDATE i DELETE i koje cine Data Manipulation Language (DML). Obrada DML instrukcija se obavlja u dvije etape : analiziranje instrukcije (*parse*) i izvrsenje instrukcije (*execute*).

U fazi analiziranja korisnicki proces odasilje instrukciju server procesu. Nakon provjere sintakse, analizirana instrukcija je ucitana u zajedicku SQL zonu (shared SQL area). Ako je faza analiziranja uspjesno obavljena, slijedeca faza u obradi instrukcije izvrsna faza.

Izvrsna faza se sastoji iz vise etapa. Posmatracemo izvsenje jedne UPDATE instrukcije koje je prikazano na slici.

Najprije, server proces cita blokove podataka iz rollback blokove iz Database Buffer Cache-a. Ukoliko zahtijevani blokovi podataka nisu pronadjeni u Database Buffer Cache-u tada ce server proces povuci te blokove podataka iz datoteka podataka (1) i smjestiti ih u Database Buffer Cache (2).

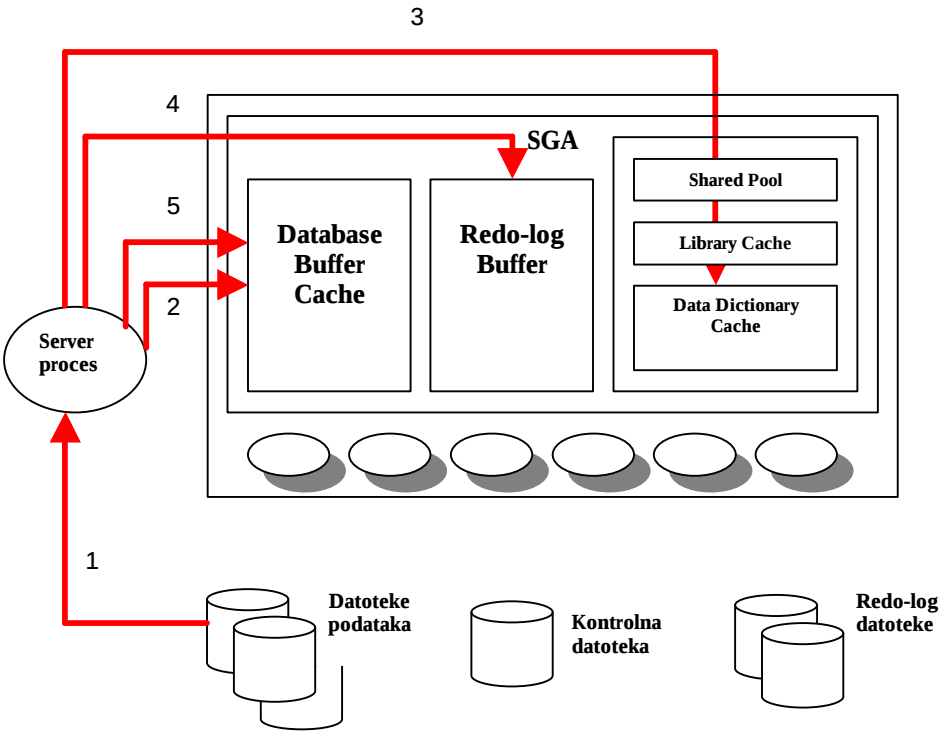
Nakon sto je smjestio zahtijevane blokove podataka u Database Buffer Cache, server proces se staviti zabrane pristupa na podatke koji trebaju da su izmjenjeni (3).

Potome, originalne vrijednosti (*before image*) podataka su smjestene u rollback segment. Ovo uzrokuje izmjene u buffer cache-u. Server proces biljezi ove izmjene u Redo-Log Buffer-u (4) stiteci tako originalne vrijednosti.

Konacno, server proces biljezi promjene u Database Buffer Cache-u (5). Ove izmjene su takodjer zabiljezene i u Redo-Log Buffer-u da bismo zastitili novu (*after-image*) vrijednost.

Izmjenjeni blokovi u database Buffer cache-u su obiljezeni kao "prljavi" medjuspremnici (*dirty bufferi*). "Prljavi" medjuspremnici su oni koji nisu identicni odgovarajucim blokovima na disku.

Obrada instrukcija DELETE i INSERT sadrzi slicne korake. Originalna vrijednost (*before image*) DELETE instrukcije sadrzi vrijednosti kolona ponistene linije dok je u slucaju INSERT-a potrebno spremiti informaciju o lokaciji linije u rollback blok.



SLIKA 1.3 shematski prikaz procesa obrade jedne DML instukcije

Rollback Segmenti

Kada pozovemo neku DML instrukciju kako bismo izmjenili podatke, server proces najprije kopira originalne, stare, vrijednosti u rollback segment. Rollback segment je koristen za ponistavanje promjena u slucaju anuliranja transakcije (*rollback*) i da bi se osiguralo da ostale transakcije ne vide nepotvrdjene (*uncommited*) izmjene.

Korisnici Oracle baze ne mogu eksplicitno citati u rollback segmentu. Jedino Oracle moze pristupati rollback segmentima.

Svaka baza podataka sadrzi jedan ili vise rollback segmenata. Oracle automatski pridruzuje transakciju slijedecem raspolozivom rollback segmentu. Do ovog pridruzivanja transakcije rollback segmlentu dolazi u momentu kada pozovete prvu DML instrukciju u vasoj transakciji.

Oracle nece nikada pridruziti transakciju koja samo cita iz baze (*read-only*) jednom rollback segmentu. Transakcije koje samo citaju iz baze ne sadrze samo upite i nijednu DML instrukciju.

Broj transakcija kojima moze raspolagati jedan rollback segment ovisi o operativnom sistemu.

Upotreba rollback segmenata

- Ako je potrebno, rollback segment je koristen prilikom anuliranja transakcije za ponistavanje izmjena nacinjenih nad podacima.
- Informacije spremljene u rollback segmentima su koristene i za osiguranje dosljednosti citanja (*read consistency*). Dosljednost citanja osigurava da ostale transakcije ne pristupaju vrijednostima koje su izmjenjene nepotvrdjenim (*uncommited*) DML instrukcijama.
- Rollback segmenti su takodjer koristeni prilikom oporavka baze nakon kvara u cilju njenog dovodjenja u dosljedno (consistent) stanje.

Rollback segmenti se nalaze u datotekama podtaka jednako kao i table i indeksi. Dijelovi ovih segmenata su po potrebi ucitavani u Database Buffer Cache.

Redo-Log Buffer

Redo-Log Buffer biljezi sve promjene nacinjene nad podacima prilikom obrade DML instrukcije.

Redo-Log Buffer je memorijska struktura koja cini dio SGA. Njegova velicina izrazena u bajtima je odredjena vrijednoscu parametra LOG_BUFFER.

U Redo-Log Bufferu su spremljeni redo zapisi ili ulazi (entries). Redo zapis biljezi izmjenjeni blok, lokaciju izmjene i novu vrijednost. Redo zapis ne pravi nikakvu razliku u pogledu tipa bloka u kome je izvrsena izmjena. Ovo znaci da redo zapis ne pravi razliku izmedju bloka podataka, indeks bloka ili rollback segment bloka.

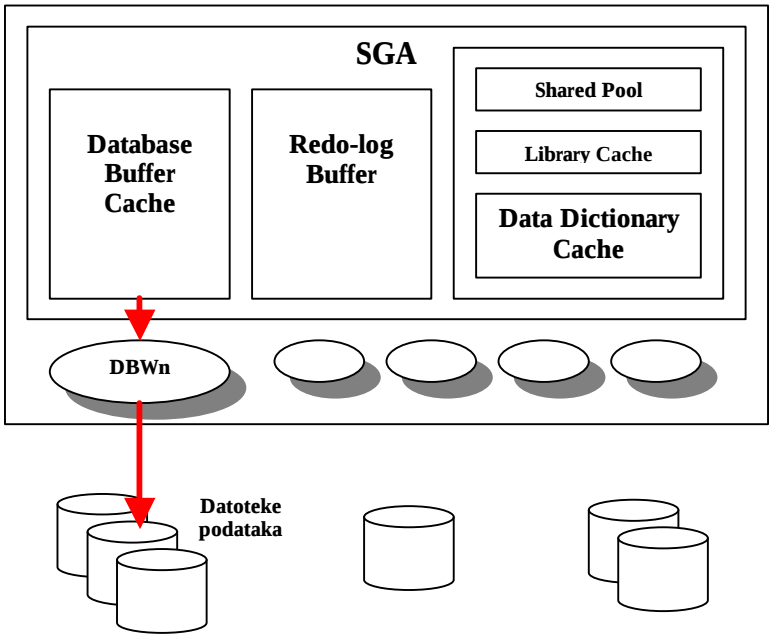
Redo-Log Buffer je koristen sekvencijalno (u slijedu) te se izmjene sacinjene u jednoj transakciji nalaze pomijesane sa izmjenama drugih transakcija.

Redo-Log Buffer je kruzni medjuspremnik (*buffer*) ciji prostor je oslobodjen za ponovnu upotrebu nakon sto je popunjen. Medjutim, do ovoga oslobadjanja prostora dolazi jedino nakon sto su svi postojeci redo zapisi upisani u online redo-log datoteke.

Velicina Redo-Log Buffera je fiksna i ona je kreirana prilikom podizanja instance.

Database Writer (DBWn)

DBWn pozadinski proces upisuje podatke iz Database Buffer Cache-a u datoteke podataka na disku. Do ovog upisivanja podataka na disk dolazi jedino kada se dese odredjeni dogadjaji.



SLIKA 1.4. shematski prikaz upisivanja podataka u datoteke podataka koje obavlja DBWn proces

- Jedan od dogadjaja koji dovode do toga da DBWn proces pocne pisati u datoteke podatake je kada broj "prljavih" medjuspremnika (*dirty buffers*) dostigne vrijednost praga (*treshold*). "Prljav" medjuspremnik je izmjenjeni medjuspremnik u Database Buffer Cache-u. Kako broj "prljavih" medjuspremnika raste tako i broj slobodnih medjuspremnika u Database Buffer Cache-u opada. Jedan od zadataka DBWn procesa je da odrzava dovoljno slobodnih medjuspremnika kako bi mogao prihvatiti ucitane blokove podataka iz datoteka podataka.

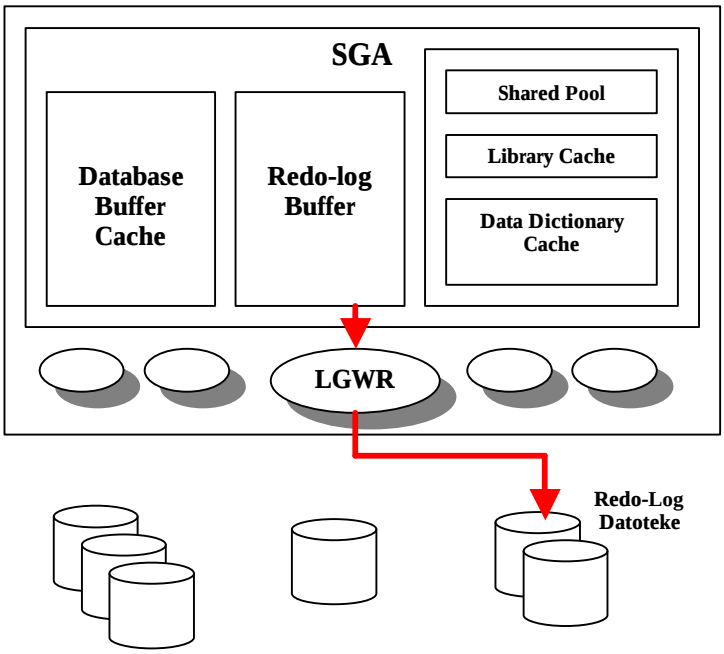
- DBWn ce poceti pisanje u datoteke podataka i kada server proces ne uspije pronaci slobodne upotrebljive medjuspremnike nakon skaniranja odredjenog broja medjuspremnika.

- Ako istekne odredjeno vrijeme timeout-a, DBWn ce odpoceti pisanje u datoteke podataka. Timeout istice ako je DBWn neaktivan tokom odredjenog vremenskog perioda, npr. tokom tri sekunde. Kada timeout istekne DBWn trazi u Last Recently Used (LRU) listi odredjeni broj medjuspremnika i upisuje "prljave" medjuspremnike na disk. Ukoliko je baza podataka nezaposlena, DBWn ce eventualno upisati na disk kompletan sadrzaj Database Buffer Cache-a.

- DBWn pise na disk i kada se desi checkpoint. Checkpoint mogu izazvati razliciti dogadjaji kao npr. kada administrator baze podataka zaustavlja bazu podataka. DBWn checkpoint je sredstvo za sinhroniziranje Database Buffer Cache-a sa datotekama podataka.

Log Writer (LGWR)

LGWR pozadinski proces upisuje redo zapise iz Redo-Log Buffera u online redo-log datoteke tekuce redo-log grupe. LGWR proces zapocinje ovo pisanje kada se desi jedan od navedenih dogadjaja :



SLIKA 1.5. shematski prikaz upisivanja podataka u redo-log datoteke koje obavlja LGWR proces

- LGWR pocinje pisanje kada je Redo-Log Buffer popunjen do trecine.
- LGWR pocinje pisanje i kada istekne timeout. Standardna vrijednost timeout-a je tri sekunde.
- LGWR ce upisati redo zapise u online redo-log datoteke prije nego sto DBWn upise bilo koji od izmjenjenih blokova podataka iz Database Buffer Cache-a u datoteke podataka.
- Kada korisnicki proces potvrdi (*commit*) transakciju, LGWR upisuje Redo-Log Buffer u online redo-log datoteke. U pojedinim situacijama kada potrebno vise prostora u Redo-Log Bufferu nego sto ga ima raspolozivog, LGWR ce upisati redo zapise prije nego sto je transakcija potvrdjena. Medjutim, ovi zapisi ce postati trajni jedino nakon sto transakcija bude potvrdjena.
- Kada je vise od 1 MB informacija upisano u Redo-Log Buffer.

LGWR proces je znacajan radi toga sto potvrda transakcije nije izdana sve dok ona nije upisana u redo-log datoteku.

Napomena: Ukoliko je jedna od datoteka iz online red-log grupe ostecena ili nedostupna, LGWR ce upisati obavijest o greski u LGWR TRACE datoteku i u sistemsku ALERT datoteku.

Obrada validiranja transakcije – instrukcija COMMIT

Kao sto je vec receno, izmjene koje izvrsite nad informacijama sadrzanim u bazi podataka trebate potvrditi kako bi one postale trajno upisane. Naravno, nista vas ne obavezuje na to, te se mozete odluciti i da anulirate pomenute izmjene. U daljem tekstu su objasnjene etape koje Oracle Server izvrsava prilikom tretmana instrukcije COMMIT.

System Change Number (SCN)

Oracle dodjeljuje System Change Number (SCN) svim potvrdjenim transakcijama. SCN je koristen za identifikovanje transakcija.

- SCN koji je dodijeljen transakciji je jedinstven za svaku transakciju u bazi podataka.
- Oracle koristi SCN kako vremenski zig prilikom sinhronizovanja podatka. SCN je logicki vremenski zig koji je koristen kako bi se stavili u poredak svi dogadjaji u jednoj instanci i kroz sve instance.
- SCN sluzi za obezbjedjivanje dosljednosti (*read consistecy*) citanja kada su podaci pribavljeni iz datoteka podataka.
- Koristeci SCN Oracle obavlja kontrolu dosljednosti neovisno o datumu i vremenu operativnog sistema.
- SCN je koristen prilikom oporavka baze podataka.

Oracle koristi razlicite sheme za generisanje SCN vrijednosti, kao sto su lock SCN shema ili Lamport SCN shema. Da biste vidjeli koju shemu Oracle koristi mozete konsultovati ALERT log datoteku nakon pokretanja instance.

Etape u obradi instrukcije COMMIT

Nakon sto je obradjena, transakcija treba da je trajno ubiljezena. Ovo je obavljeno izdavanjem naredbe COMMIT. Nakon sto je izdana komanda COMMIT Oracle izvrsava niz koraka.

Najprije, server proces postavlja u Redo-Log Buffer commit zapis zajedno sa njegovim SCN brojem (1).

Potom LGWR obavlja (*contiguous*) upisivanje svih redo zapisa iz Redo-Log Buffer-a od prvog do zadnjeg, ukljucujuci i commit zapis, u redo-log datoteke (2). Nakon ove tacke, sigurni ste da promjene nece biti izgubljene cak ni u slucaju kvara.

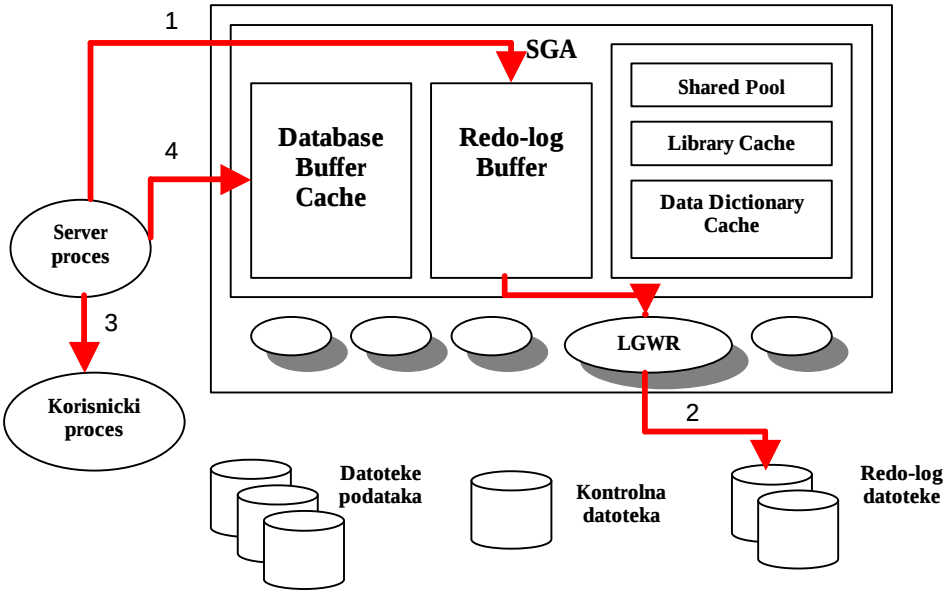
Slijedeci korak je proslijedjivanje informacije korisniku o zavrsetku commit-a (3).

Konacno, server proces biljezi informaciju koja ukazuje na kraj transakcije (4). Zabrane pristupa postavljene nad tablama ili linijama su podignute.

Na ovaj nacin je zavrsen proces COMMIT. Proces COMMIT koji je izvrsen na ovakav nacin nazivamo brzi (*fast*) COMMIT. Primjeticete da je samo Redo-Log Buffer upisan na disk.

Prebacivanje (*flushing*) "prljavih" medjuspremnika iz Database Buffer cache-a u datoteke podataka na disku obavlja proces DBWn neovisno o commit procesu i moze da bude obavljeno prije ili pisloje commit-a.

Samo jedno pisanje je potrebno za svaki commit proces. Ako vise korisnika simultano izdaju commit naredbu, Oracle ce strpati (*piggyback*) sve commit-e u jedno pisanje. Ovo omogucuje da se postigne manje od jednog I/O po commit-u na aktivnim sistemima.



SLIKA 1.5. shematski prikaz etapa u obradi COMMIT instrukcije

Prednosti obrade brzog COMMIT-a

Oracle koristi mehanizam brzog COMMIT-a kako bi osigurao oporavak baze u slucaju kvara. Prednosti brzog COMMIT-a su slijedece :

- Brzi COMMIT obavlja sekvencijalne upise u redo-log datoteke umjesto da pise u datoteke podataka na disku. Ovo je znatno brze od upisivanja podataka u razlicite blokove u datokema podataka.

- Brzi COMMIT pise samo podatke neophodne za biljezenje promjena u redo-log datotekama. Ovo predstavlja ustedu u vremenu obzirom da pisanje u datoteke podataka zahtijeva upisivanje citavih blokova podataka a ne samo izmjenjenog dijela.

- Koristeci brzi COMMIT, baza podataka ujedinjuje (*piggyback*) zahtjeve za commit-om. Na ovaj nacin je omoguceno da vise transakcija obave commit u isto vrijeme u samo jednom pisanju (I/O pristupu disku).

- Brzi COMMIT zahtijeva samo jedno sinhrono pisanje po transakciji. Ovo je tacno sve dok se Redo-Log Buffer ne napuni. U tome slucaju dodatni I/O mogu biti potrebni.

Velicina transakcije ne utice na vrijeme potrebno za COMMIT operaciju.