

Knjiga

1

FREE ANONIMA – SAMOSTALNA OBUKA ORACLE

---

<http://stoga.anonima.free.fr>

# Oracle8i DBA Arhitektura i administriranje

ORACLE8I DBA (1Z0-023)

# Arhitektura i administriranje Oracle8i servera baze podataka

---

©2002 Informacije sadržane u ovoj knjizi su isključivo vlasništvo Free @nonime. Svako umnozavanje, djelimicno ili potpuno, kao i komercijalna eksploatacija sadržaja knjige su zabranjeni.

---

# Sadržaj

<hr/>		Etape u obradi instrukcije COMMIT	20
<b>Uvod u Oracle Server</b>		Prednosti obrade brzog COMMIT-a	22
Zadaci administratora baze podataka	1		
Prikljucivanje na Oracle Server	2		
Korisnicki procesi	4		
Server procesi	4		
<hr/>			
<b>Oracle instanca i baza podataka</b>			
Sistem Global Area (SGA)	5		
Pozadinski procesi	6		
Fizicka struktura (datoteke) baze podataka	7		
Ostale datoteke	9		
<hr/>			
<b>Obrada upita u bazi podataka</b>			
Etape obrade upita u bazi podataka	10		
Analiziranje upita	10		
Izvršenje upita	11		
Dobavljanje	11		
Shared Pool	11		
Database Buffer Cache	12		
Program Global Area (PGA)	14		
<hr/>			
<b>Obrada DML instrukcija</b>			
Izvršna faza DML instrukcije	15		
Rollback Segmenti	16		
Redo-Log Buffer	17		
Database Writer (DBWn)	17		
Log Writer (LGWR)	19		
<hr/>			
<b>Obrada validiranja transakcije, instrukcija COMMIT</b>			
System Change Number (SCN)	20		

---

# Arhitekturne komponente Oracle servera baze podataka

*U ovome poglavlju su prikazane osnovne arhitekturne komponente jednog ORACLE servera baze podataka. Kao budući administratori baze podataka trebate već u samom startu da ovladate ovim osnovnim pojmovima te da se upoznate sa zadacima koje ćete obavljati u toj funkciji. Potpuno razumijevanje ovih osnovnih komponenti servera Oracle i njihovih međusobnih relacija je neophodno za dalji rad.*

## Uvod u Oracle Server

**O**racle Server je baza podataka koju odlikuje visoka raspoloživost i efikasnost pri radu sa značajnim brojem korisnika i velikim obimom informacija. Oracle Server je ključni element u izradi informacionih sistema koji sadrži sve podatke neophodne za pravovremeno i efikasno djelovanje preduzeća. Ovakvo centralno mjesto baze podataka u jednom informacionom sistemu podrazumijeva njeno efikasno i redovno održavanje. Kao što ćete vidjeti u daljem tekstu, osiguravanje raspoloživosti, korektnog i efikasnog funkcionisanja baze podataka su zadatak administratora.

## Zadaci administratora baze podataka

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

- Jedan od najvažnijih zadataka administratora baze podataka je instaliranje Oracle servera i aplikativnih alata. Nakon instaliranja Oracle servera, administrator baze podataka je također dužan 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 buduće 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 postojećim, 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 izvršava 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 priključak, korisnik se može direktno prijaviti na masinu na kojoj je instaliran Oracle server. Npr. korisnik se priključuje na UNIX masinu na kojoj je instaliran Oracle server i pomoću alata Server Manager pristupa bazi podataka.

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

U troslojnom priključku masina korisnika baze podataka je priključena na mrežni (aplikativni) server. Ovaj mrežni server je putem mreže priključen na masinu na kojoj je instaliran Oracle server. Primjer troslojnog priključka je kada korisnik baze podataka pomoću internet navigatora na svojoj masini startuje aplikaciju na NT serveru. Server NT potom pribavlja podatke iz baze na UNIX serveru.

Kada se korisnik priključuje na masinu na kojoj funkcioniše Oracle server, dva procesa su uključena, jedan korisnički i jedan server proces.

Korisnički proces je kreiran kada korisnik pokrene alat kao što je SQL\*Plus ili aplikaciju izradjenu pomoću alata kao što je npr. Oracle Forms. Korisnički proces je mehanizam koji služi za izvršavanje koda aplikativnog programa kao što je Oracle Forms aplikacija ili nekog Oracle alata kao što je OEM. U klijent/server modelu Oracle alat ili aplikacija se izvršavaju na klijent masini.

Server proces je kreiran kada se korisnik prijavi na Oracle server precizirajući svoje ime korisnika, lozinku i ime baze podataka. Server proces je kreiran na istoj masini na kojoj funkcioniše Oracle server. Server proces je mehanizam interakcije između Oracle servera i korisničkog procesa na masini klijentu. Server proces obavlja na Oracle serveru operacije koje zahtijeva korisnički proces.

Komunikacijski put između korisničkog procesa i Oracle servera nazivamo priključkom. Kada korisnik i Oracle server djeluju međusobno priključak je uspostavljen.

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

Ako korisnik koristi alat ili aplikaciju na klijent masini, za komunikaciju između korisnika i Oracle servera putem mreže neophodna je upotreba mrežnog softvera.

Zaseban priključak korisnika na Oracle server nazivamo sesijom. Sesija počinje od momenta kada Oracle server potvrdi identitet korisnika a završava se odjavljivanjem korisnika ili nenormalno.

Više istovremenih sesija može postojati za korisnika koji se priključi na Oracle server sa različitim terminala.

## Korisnicki procesi

Da bi izvršio aplikativni alat kao što je SQL\*Plus ili aplikaciju uradjenu pomoću alata kao što je Oracle Forms, Oracle kreira korisnicki proces. Korisnicki proces se također naziva i klijentom. Ovaj proces je izvršavan na masini na kojoj se korisnik direktno prijavljuje.

Korisnicki proces je pokrenut kada korisnik startuje alat a završava se odjavljivanjem korisnika ili uslijed nenormalnog završetka rada kao što je pad sistema.

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

UPI je također 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 što je zahtjev obradjen na Oracle serveru, rezultat je uzvracen korisnickom procesu.

## Server procesi

Kao što smo vidjeli, korisnicki proces prosljedjuje 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 završava 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 zahtjev korisnickog procesa.

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

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

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

## Oracle instanca i baza podataka

Globalno uzevši 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.

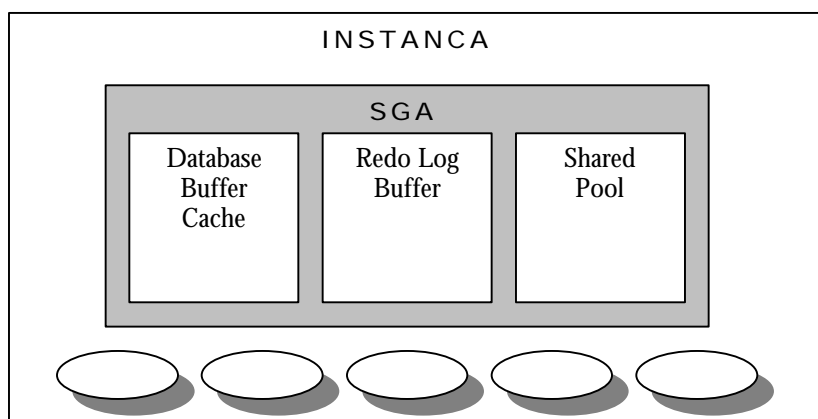
Jedan od razloga za ovu konfuziju je i taj da, u vecini slucajeva, postoji relacija jedan-na-jedan izmedju ova dva pojma. Ovo znaci da uvijek, osim u slucaju Oracle Parallel Servera, jedna instanca pristupa jednoj bazi podataka. U slucaju izuzetka koji je Oracle Parallel Server, vise instanci pristupaju 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 služi za pohranjivanje informacija kao što su najskorije izvršene SQL instrukcije i skoro korištene informacije rječnika podataka.



Database Buffer Cache poznat kao Data Buffer Cache služi za pohranjivanje najskorije korištenih podataka korisnika. On sadrži u memoriji kopije blokova podataka radi postizanja efikasnog funkcionisanja baze.

Redo Log Buffer poznat i kao Log Buffer Cache služi za pohranjivanje promjena izvršenih u bazi podataka.

Memorijska zona SGA je read-write, što znači da je moguće iz nje citati i u nju upisivati podatke. Bilo ko priključen na bazu podataka može citati u SGA, dok jedino skup pozadinskih procesa može 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 zahtjeva koji dolaze istovremeno od strane više korisnika.

Pozadinski procesi su procesi operativnog sistema u većini operativnih sistema osim u Windows NT gdje se radi o thread-ima jednog procesa. Svaka instanca može koristiti više pozadinskih procesa čiji 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 više instanci, SMON proces jedne instance obavlja oporavak drugih instanci koje su pretrpile pad. Pored ovoga, zadatak SMON procesa je i da obavi čiscenje privremenih segmenata koji više nisu u upotrebi te da oporavi « mrtve » transakcije ispustene prilikom pada sistema do kojeg je doslo uslijed neke greske. SMON također ujedinjuje fragmentirani slobodni prostor u datotekama podataka.

Process Monitor (PMON) proces obavlja oporavak (*recovery*) procesa kada neki od korisničkih procesa pretrpi neuspjeh. PMON proces je odgovoran za čiscenje cache memorije i oslobađanje resursa sistema koje je držao korisnički proces.

Checkpoint (CKPT) proces je odgovoran za azuriranje informacije o statusu baze podataka, kao što 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 događaja ili nakon log switch-a (izmjene aktivne redo-log datoteke).

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

Recoverer (RECO) proces je koristen za rjesavanje 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 služi 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 dispatcher procesa izmedju D000 i Dnnn.

## Fizicka struktura (datoteke) baze podataka

Kao što je već rečeno Oracle Server posjeduje dvije osnovne komponente: bazu podataka i instancu. Jako često su ova dva pojma zamjenjivana ili pogrešno upotrebljavana. Donekle je ova zbrka uzrokovana odnosom jedan-na-jedan koji postoji u većini slučajeva izmedju jedne Oracle baze podataka i Oracle instance. Drugim riječima, 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 služi da pohranjujemo i ponovno pronalazimo medjusobno povezane informacije. Ona se sastoji od fizickih struktura skladištenja, 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 postojećim 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 učitavani, svaki put kada je to zahtijevano i spremjeni u u memorijsku strukturu nazvanu Database Buffer Cache. Izmjenjeni i novi podaci nisu trenutno upisivani u datoteke podataka. U cilju efikasnijeg funkcionisanja baze podataka oni su spremeni 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 poboljšava efikasnost rada baze podataka.

### Redo-log datoteke

Redo-log datoteke su koristene za bilježenje promjena izvršenih nad podacima (instrukcijama insert, update ili delete) i osiguravanje oporavka podataka u slučaju pada instance. Svaka redo stavka (*redo entry*) sadrži 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 zabilježene u redo-log datotekama, svaki kvar je moguće opraviti koristeći 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 zaštitili od gubljenja redo-log datoteka uslijed kvarova na disku Oracle nam omogućuje umnozavanje (multipleksiranje) redo-log datoteka. U ovom slučaju moguće je držati dvije ili više redo-log datoteka na različitim diskovima. Kopije redo-log datoteka držane na različitim diskovima nazivamo *mirror log* datotekama. Grupa multipleksiranih redo-log datoteka naziva se redo-log grupom. Svaki član u redo-log grupi ima istu veličinu i informacije su istovremeno u njih upisivane.

Interes držanja redo-log datoteka na različitim diskovima leži u tome da u slučaju kvara na jednom disku možemo i dalje raspolagati datotekama na preostalim diskovima.

### Kontrolne datoteke

Kontrolne datoteke sadrže informacije neophodne za održavanje i provjeru integriteta baze podataka. One sadrže strukturu baze podataka, kao što su ime baze podataka, vremenski žig (*timestamp*) kreiranja baze podataka, imena i lokacije datoteka podataka i redo-log datoteka kao i njihovo stanje (online ili offline). Kontrolne datoteke sadrže također i log broj trenutno koristene redo-log datoteke.

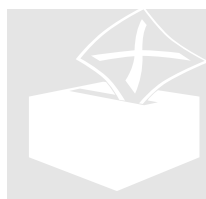
Informacije pohranjene u kontrolnim datotekama su koristene za identifikovanje datoteka podataka i redo-log datoteka koje moraju biti otvorene kada otvaramo bazu podataka. Ukoliko je struktura baze podataka izmjenjena Oracle automatski mijenja kontrolne datoteke kako bi one odrazile ove izmjene.

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

kontrolna datoteka može biti korištena i za provjeru konzistentnosti datoteka podataka prilikom pokretanja baze podataka.

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

Velicina kontrolne datoteke je promjenjiva i ovisi o definisanim vrijednostima pojedinih ograničenja 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 sačinjavaju Oracle bazu podataka. Iako Oracle koristi i druge važne datoteke, kao što 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 što su podizanje instance ili identifikovanje korisnika. Ove datoteke su drugi vid strukture fizičkog pohranjivanja koje koristi Oracle server. Postoje tri tipa ovih datoteka : datoteka parametara, datoteka lozinki i arhivirane redo-log datoteke. Oracle također generise ALERT i TRACE datoteke. To su datoteke u koje Oracle upisuje izvještaje o statusu i greškama. ALERT datoteke pružaju važne informacije o statusu instance i proizvedenim greškama.

### Datoteka parametara

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

Inicijalizacijski parametri definišu ime baze podataka, količinu memorije koju treba rezervirati za instancu, imena kontrolnih datoteka i brojne druge parametre.

### Datoteka lozinki

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

### Arhivirane redo-log datoteke

Arhivirane redo-log datoteke služe 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 što su primjenjene redo-log datoteke, rollback segmenti su korišteni za identifikovanje i anuliranje nepotvrđenih transakcija ubilježenih u redo-log. Ovaj proces anuliranja nepotvrđenih transakcija naziva se *roll back*.

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

Ako baza podataka funkcionise u ARCHIVELOG načinu rada, redo-log datoteke su arhivirane prije nego što su nanovo korištene u ciklicnom slijedu. Ako, naprotiv, baza podataka funkcionise u NOARCHIVELOG načinu rada, redo-log datoteke nisu arhivirane prije nego što su ponovo korištene (njihov sadržaj je pogazen novim upisima).

U ARCHIVELOG načinu rada baza podataka može biti potpuno oporavljena i nakon kvara instance i nakon kvara medija.

## Obrada upita u bazi podataka

Sigurno ste već imali priliku koristiti alat kao što je na primjer SQL\*Plus u cilju pribavljanja informacija iz Oracle baze podataka. Pomenuta aplikacija vam pruža mogućnost da jednostavno odasijete zahtijev Oracle Serveru koristeći se pri tome SQL instrukcijama koje pisete u editoru teksta aplikacije i potom ih izvršavate jednostavnim pritiskom na tipku Enter. Međutim, od momenta kada ste potvrdili vas zahtijev tipkom Enter do trenutka kada vam Oracle Server uzvratiti rezultat koji odgovara vašem zahtijevu desava se iza kulisa citav niz operacija. Jedan bezazleni zahtijev tipa « zelim znati imena radnika zaposlenih u toku ove godine » će uzrokovati niz interakcija različitih arhitekturnih komponenti Oracle baze podataka kao što je detaljno predstavljeno u daljem tekstu.

### Etape obrade upita u bazi podataka

Kada korisnik postavi jednu SQL instrukciju ili upit (query), korisnički proces prosljeđuje 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*), izvršenje (*execute*) i dobavljanje (*fetch*).

#### Analiziranje upita

U ovoj etapi server proces najprije prima upit od korisničkog 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.

### Izvršenje 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.

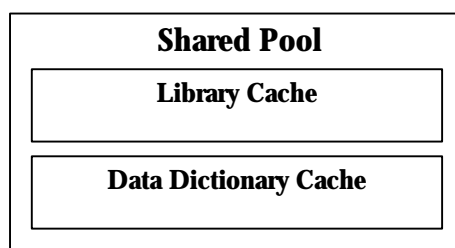
### Dobavljanje

U ovoj fazi server povlaci linije koji se zahjev tice i prosljedjuje 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 korištenim SQL instrukcijama. U njoj su pohranjeni tekst instrukcije, kompilirana verzija instrukcije (*parse tree*) te plan izvršenja instrukcije (*execution plan*). Plan izvršenja komande sadrzi korake izvršenja instrukcije koje određuje optimizator.

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

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

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

Data Dictionary Cache sadrzi informacije rječnika 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 sadržani u linijama a ne u medjuspremnicima (*buffer*) koji sadrže 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 sadržava blokove podataka koji su učitani iz datoteka podataka. Ukoliko server proces ne uspije pronaci zahtijevani blok podataka u Database Buffer Cache-u, on ce taj blok podataka učitati iz datoteke podataka i njegovu kopiju spremi u Database Buffer Cache.

Sa stanovista efikasnosti rada baze podataka, poželjno je da server proces pronalazi potrebne podatke u memoriji (*cache hit*). Ukoliko to nije slučaj (*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 medjuspremnik (*buffer-a*) baze podataka. Velicina svakog od medjuspremnik u Database Buffer Cache-u je jednaka velicini jednog bloka podataka u bazi podataka.

Velicina memorijske zone Database Buffer Cache jednaka je proizvodu broja medjuspremnik (*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 medjuspremnik (*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  $DB\_BLOCK\_BUFFERS$  dinamički parametar, da bismo izmjenili velicinu Database Buffer-a neophodno je zaustaviti i nanovo podici instancu.

Database Buffer Cache sadrži 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 medjuspremnicki sa podacima koji nisu skoro bili korišteni.

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

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

LRU lista sadrži medjuspremnicke koji mogu da su ili trenutno korišteni ili slobodni za upotrebu ali isto tako i modifikovani medjuspremnicki koji još nisu prebaceni u necistu listu. Do njihovog premještanja u necistu listu će doći kada neki drugi proces bude tražio slobodne medjuspremnicke u LRU listi.

Kada jedan proces nastoji da rezervise medjuspremnicke u Database Buffer-u, on će tražiti slobodne medjuspremnicke u LRU listi. Taj proces će skanirati LRU listu dok ne pronađe slobodne medjuspremnicke ili dok ne bude dostignut određeni prag. Prilikom ovog skaniranja svi pronađeni izmjenjeni medjuspremnicki će biti prebaceni u necistu listu. Ukoliko slobodni medjuspremnicki nisu pronađeni proces će to signalirati DBWn procesu kako bi on reagovao tako što će upisati izmjenjene medjuspremnicke na disk i na taj način osloboditi prostor u Database Buffer-u.

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

U Oracle8i bazi raspolazemo sa više zona u Database Buffer Cache-u. Objekte možemo učitati u jednu od slijedeće tri zone:

- DEFAULT zona je korištena za objekte koji nisu pridruženi određenoj zoni Database Buffer-a ili su, što je prilično rijetko, eksplicitno pridruženi DEFAULT zoni. Velicina ove zone nije eksplicitno određenaveć je za nju rezervisan prostor koji preostane u Database Buffer-u nakon rezervisanja prostora za preostale dvije zone: KEEP i RECYCLE.
- KEEP zona je korištena za objekte koje želimo zadržati u memoriji. Njena velicina je kontrolisana BUFFER\_POOL\_KEEP parametrom.
- RECYCLE zona je korištena za objekte koje želimo zadržati 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 pojedinačni pozadinski proces.

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

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

Sadržaj 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 što joj ime kaže, korištena je za obavljanje operacija razvrstavanje (*sort*), koje mogu biti neophodne prije obrade linija ili prije nego što su linije ođaslane korisničkom procesu.

Informacije o sesiji preciziraju privilegije korisnika za tekucu sesiju.

Stanje kursora ukazuje na fazu u kojoj se nalazi obrada pojedinih kursora koji su trenutno korišteni u sesiji.

Kursor je poanter ka memorijskoj zoni pridruženoj specifičnoj SQL instrukciji. Oracle koristi radne zone za izvršavanje SQL instrukcija i spremanje informacija o obradi. Kursori nam omogućuju da imenujemo ove radne zone i da pristupamo informacijama koje one sadrže.

Stack Space sadrži varijable i matrice (tablice) sesije.

## Obrada DML instrukcija

**P**ored instrukcije SELECT kojom pribavljamo informacije iz Oracle baze podataka na raspolaganju su nam i instrukcije koje nam omogućuju modifikovanje sadržaja tih informacija. Pod modifikovanjem podrazumijevamo kako izmjenu sadržaja informacija (azuriranje) tako i dodavanje novih informacija te brisanje odnosno ponistavanje postojećih informacija. Tretman ovih instrukcija koje još nazivamo i DML (Data Manipulation Language) instrukcijama razlikuje se od prethodno prikazanog tretmana instrukcija SELECT i uključuje i potrebu validiranja ili ponistavanja sacinjenih izmjena.

### Izvršna faza DML instrukcije

Oracle raspolaze instrukcijama za manipulisanje podataka kao što su INSERT, UPDATE i DELETE i koje cine Data Manipulation Language (DML). Obrada DML instrukcija se obavlja u dvije etape : analiziranje instrukcije (*parse*) i izvršenje 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 uspješno obavljena, slijedeca faza u obradi instrukcije izvršna faza.

Izvršna faza se sastoji iz više etapa. Posmatracemo izvršenje 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).

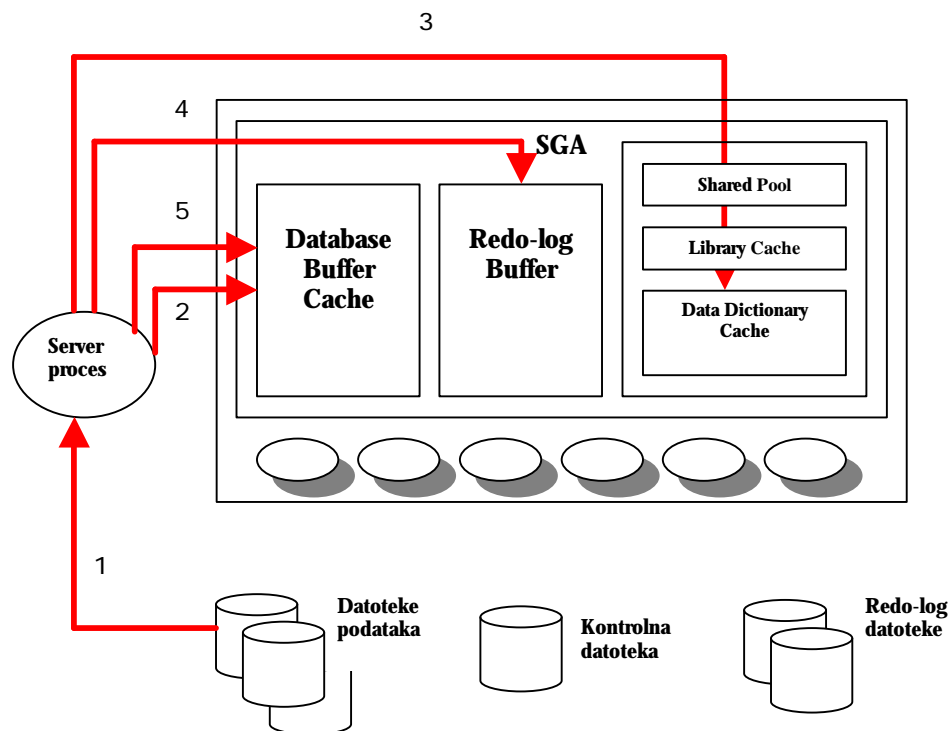
Nakon što je smjestio zahtijevane blokove podataka u Database Buffer Cache, server proces se stavi zabrane pristupa na podatke koji trebaju da su izmjenjeni (3).

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 obilježeni kao “prljavi” medjuspremnicki (*dirty buffers*). “Prljavi” medjuspremnicki su oni koji nisu identicni odgovarajucim blokovima na disku.

Obrada instrukcija DELETE i INSERT sadrži slične korake. Originalna vrijednost (*before image*) DELETE instrukcije sadrži vrijednosti kolona poništene linije dok je u slučaju INSERT-a potrebno spremići informaciju o lokaciji linije u rollback blok.



SLIKA 1.3 shematski prikaz procesa obrade jedne DML instrukcije

## Rollback Segmenti

Kada pozovemo neku DML instrukciju kako bismo izmjenili podatke, server proces najprije kopira originalne, stare, vrijednosti u rollback segment. Rollback segment je koristan za poništavanje promjena u slučaju anuliranja transakcije (*rollback*) i da bi se osiguralo da ostale transakcije ne vide nepotvrđene (*uncommitted*) izmjene.

Korisnici Oracle baze ne mogu eksplicitno citati u rollback segmentu. Jedino Oracle može pristupiti rollback segmentima.

Svaka baza podataka sadrži jedan ili više rollback segmenata. Oracle automatski pridružuje transakciju slijedecem raspoloživom rollback segmentu. Do ovog pridruživanja transakcije rollback segmentu dolazi u momentu kada pozovete prvu DML instrukciju u vašoj transakciji.

Oracle neće nikada pridružiti transakciju koja samo čita iz baze (*read-only*) jednom rollback segmentu. Transakcije koje samo čitaju iz baze ne sadrže samo upite i nijednu DML instrukciju.

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

Upotreba rollback segmenata

- Ako je potrebno, rollback segment je korišten 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 nepotvrđenim (*uncommitted*) DML instrukcijama.
- Rollback segmenti su takodjer korišteni prilikom oporavka baze nakon kvara u cilju njenog dovodjenja u dosljedno (*consistent*) stanje.

Rollback segmenti se nalaze u datotekama podataka 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 izvršena izmjena. Ovo znaci da redo zapis ne pravi razliku izmedju bloka podataka, indeks bloka ili rollback segment bloka.

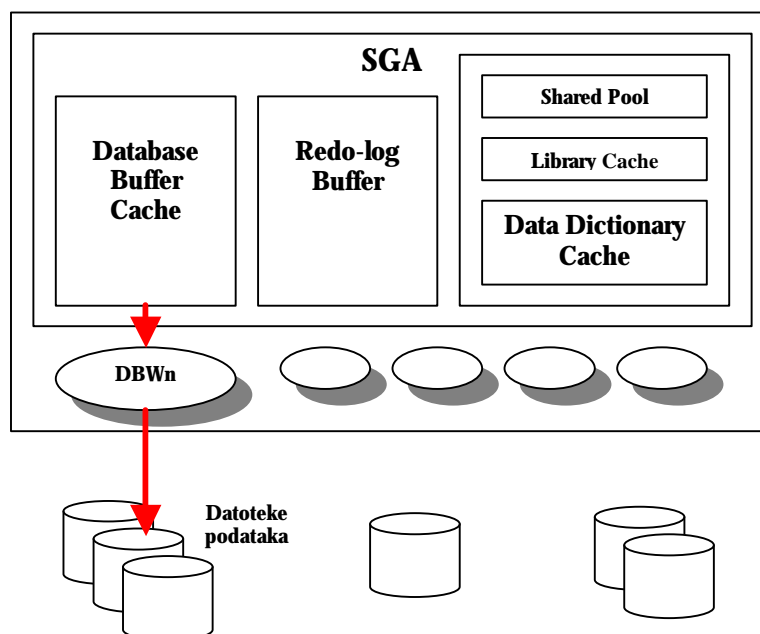
Redo-Log Buffer je korišten sekvencijalno (u slijedu) te se izmjene sacinjene u jednoj transakciji nalaze pomijesane sa izmjenama drugih transakcija.

Redo-Log Buffer je kružni 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.

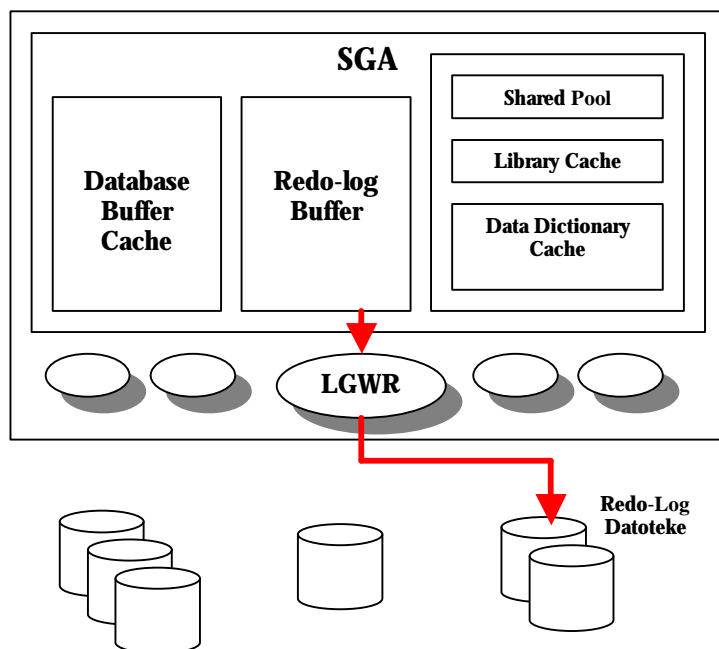


SLIKA 1.4. shematski prikaz upisivanja podataka u datoteke podataka koje obavlja DBWn proces

- Jedan od događaja koji dovode do toga da DBWn proces počne pisati u datoteke podataka je kada broj “prljavih” međuspremnik (dirty buffers) dostigne vrijednost praga (threshold). “Prljav” međuspremnik je izmjenjeni međuspremnik u Database Buffer Cache-u. Kako broj “prljavih” međuspremnik raste tako i broj slobodnih međuspremnik u Database Buffer Cache-u opada. Jedan od zadataka DBWn procesa je da održava dovoljno slobodnih međuspremnik kako bi mogao prihvatiti učitane blokove podataka iz datoteka podataka.
- DBWn će početi pisanje u datoteke podataka i kada server proces ne uspije pronaci slobodne upotrebljive međuspremnik nakon skaniranja određenog broja međuspremnik.
- Ako istekne određeno vrijeme timeout-a, DBWn će odpočeti pisanje u datoteke podataka. Timeout istice ako je DBWn neaktivan tokom određenog vremenskog perioda, npr. tokom tri sekunde. Kada timeout istekne DBWn traži u Last Recently Used (LRU) listi određeni broj međuspremnik i upisuje “prljave” međuspremnike na disk. Ukoliko je baza podataka nezaposlena, DBWn će eventualno upisati na disk kompletan sadržaj Database Buffer Cache-a.
- DBWn piše na disk i kada se desi checkpoint. Checkpoint mogu izazvati različiti događaji 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)

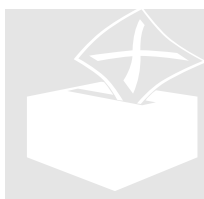
LGWR pozadinski proces upisuje redo zapise iz Redo-Log Buffera u online redo-log datoteke tekuće redo-log grupe. LGWR proces zapocinje ovo pisanje kada se desi jedan od navedenih dogadjaja :



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

- LGWR pocinje pisanje kada je Redo-Log Buffer potpunjen 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 potvrđjena. Medjutim, ovi zapisi ce postati trajni jedino nakon sto transakcija bude potvrđjena.
- 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 oštećena ili nedostupna, LGWR će upisati obavijest o greski u LGWR TRACE datoteku i u sistemsku ALERT datoteku.

## Obrada validiranja transakcije –instrukcija COMMIT

**K**ao što je već rečeno, izmjene koje izvršite nad informacijama sadržanim u bazi podataka trebate potvrditi kako bi one postale trajno upisane. Naravno, nista vas ne obavezuje na to, te se možete odluciti i da anulirate pomenute izmjene. U daljem tekstu su objasnjene etape koje Oracle Server izvršava prilikom tretmana instrukcije COMMIT.

### System Change Number (SCN)

Oracle dodjeljuje System Change Number (SCN) svim potvrđenim 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 služi 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 što su lock SCN shema ili Lamport SCN shema. Da biste vidjeli koju shemu Oracle koristi možete konsultovati ALERT log datoteku nakon pokretanja instance.

### Etape u obradi instrukcije COMMIT

Nakon što je obradjena, transakcija treba da je trajno ubilježena. Ovo je obavljeno izdavanjem naredbe COMMIT. Nakon što je izdana komanda COMMIT Oracle izvršava 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, uključujući i commit zapis, u redo-log datoteke (2). Nakon ove tace, sigurni ste da promjene neće biti izgubljene čak ni u slučaju kvara.

Slijedeci korak je prosljedjivanje informacije korisniku o završetku commit-a (3).

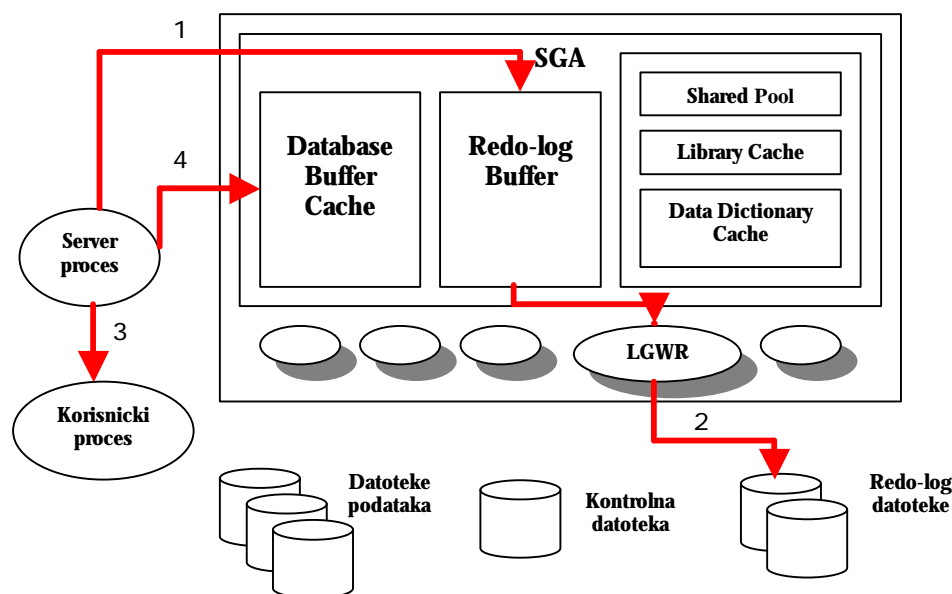
Konacno, server proces biljezi informaciju koja ukazuje na kraj transakcije (4). Zabrane pristupa postavljene nad tablama ili linijama su podignute.

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

Prebacivanje (*flushing*) “prljavih” medjusprennika iz Database Buffer cache-a u datoteke podataka na disku obavlja proces DBWn neovisno o commit procesu i može da bude obavljeno prije ili poslije commit-a.

Samo jedno pisanje je potrebno za svaki commit proces. Ako vise korisnika simultano izdaju commit naredbu, Oracle će strpati (*piggyback*) sve commit-e u jedno pisanje. Ovo omogućuje 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 slučaju kvara. Prednosti brzog COMMIT-a su slijedeće :

- Brzi COMMIT obavlja sekvencijalne upise u redo-log datoteke umjesto da piše u datoteke podataka na disku. Ovo je znatno brže od upisivanja podataka u različite blokove u datotekama podataka.
- Brzi COMMIT piše samo podatke neophodne za bilježenje promjena u redo-log datotekama. Ovo predstavlja uštedu 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 način je omogućeno da više 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 tačno sve dok se Redo-Log Buffer ne napuni. U tome slučaju dodatni I/O mogu biti potrebni.

Velicina transakcije ne utiče na vrijeme potrebno za COMMIT operaciju.