SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Seminarski rad iz predmeta Sigurnost računalnih sustava

Sigurnost podataka u Oracle bazi

Vedran Vrbanić

Zagreb, veljača 2009.

Sadržaj

Uvod 1

1. Arhitektura i dizajn baze 2

1.1. Memorijske strukture 2

1.1.1. *Buffer cache* 3

1.1.2. *Log buffer* 3

1.1.3. *Shared pool* 4

1.2. Fizička struktura podataka 5

1.2.1. Kontrolna datoteka 5

1.2.2. *Redo log* datoteke 6

1.2.3. Podatkovne datoteke 6

1.3. Logička struktura podataka 7

2. Administracija korisnika 9

2.1. Dodjeljivanje prava korisnicima 9

2.2. Role 9

2.3. Profili 11

3. Konkurentno izvođenje transakcija 13

3.1. ACID koncept 13

3.2. Implementacija ACID koncepta 13

4. Izrada sigurnosnih kopija i oporavak baze 15

4.1. Prekid rada instance 15

4.2. Kvar medija za pohranu podataka 16

5. Zaključak 18

Literatura 19

Uvod

Velika količina podataka u poslovanju i potreba za brzom ekstrakcijom informacija i znanja mijenjaju paradigme poslovanja, a informacija postaje najjače oružje u borbi sa konkurencijom. Informacijski sustav, odnosno baza podatka koja pruža pravovremene i konzistentne podatke predstavlja konkurentnu prednost u tržišnom nadmetanju. Stoga takav sustav mora zadovoljavati visoke kriterije koji se tiču sigurnosti i raspoloživosti podataka, kontrole pristupa, konkurentnog izvođenja upita i transformacija te oporavka u slučaju prekida rada baze ili kvara na medijima za pohranu podataka. U ovom su seminarskom radu objašnjene tehnike i koncepti kojima se navedeni zahtjevi ostvaruju u najnovijoj verziji *Oracle* baze: *Oracle Database 11g*. U prvom je poglavlju opisana arhitektura i dizajn baze na fizičkom i logičkom nivou. Tu su obuhvaćene memorijske strukture u bazi, tri najvažnije vrste datoteka i njihove funkcionalnosti te odnosi logičkih i fizičkih struktura u bazi podataka. Nakon toga slijedi prikaz problema i rješenja vezanih uz kontrolu pristupa bazi, dodjeljivanje privilegija nad objektima i podacima te ograničavanje resursa korisnicima. Treće poglavlje obrađuje konkurentno izvođenje upita te implementaciju atomarnosti, konzistentnosti, integriteta i trajnosti podataka. Izrada sigurnosnih kopija je obrađena u posljednjem poglavlju, skupa s tehnikama oporavka od kvarova uzrokovanih iznenadnim prekidom rada instance ili oštećenjem medija za pohranu podataka.

1. Arhitektura i dizajn baze

Oracle baza sastoji se od datoteka pohranjenih na disku te memorijskih struktura i procesa koji se izvršavaju na baznom serveru. Memorijske strukture i procesi čine instancu baze. Prvi je korak prilikom pokretanja baze izgradnja instance u memoriji. Nakon toga, instanca dohvaća i čita datoteke na disku. Memorijske strukture koje čine instancu pohranjene su u SGA (System Global Area) memorijskom segmentu. SGA se alocira pri pokretanju instance. Sastoji se od nekoliko komponenata čija se veličina može dinamički mijenjati dok je instanca aktivna. Korisničke aplikacije uspostavljaju sesije s bazom. Sesija se sastoji od korisničkog procesa (aplikacije) koji se izvršava na lokalnom računalu i serverskog procesa koji se izvršava na serveru. Korisnički proces generira SQL naredbe, a serverski ih proces izvršava. Svakoj je sesiji pridružen jedan serverski proces. Svaki takav proces zauzima dio radne memorije za vlastite potrebe. Ta se memorija naziva PGA (Program Global Area). PGA je memorija koju koristi isključivo jedan serverski proces, dok je SGA na raspolaganju svim serverskim procesima. Podatkovne se strukture mogu promatrati s dva aspekta – logičkog i fizičkog. Podaci su u logičkom smislu pohranjeni u segmentima. Postoje različiti tipovi segmenata, a najznačajniji su tablice i indeksi. Segmenti su fizički pohranjeni u podatkovnim datotekama na disku. Veze između fizičkih i logičkih struktura podataka pohranjene su u rječniku podataka.

1.1. Memorijske strukture

SGA memorijski segment se sastoji od 6 manjih segmenata:

- *Buffer cache*
- *Log buffer*
- *Shared pool*
- *Large pool*
- *Java pool*
- *Streams pool*

Prva su tri segmenta obavezna, dok su posljednja tri opcionalna. Upravljanje veličinom ovih

struktura može biti automatsko, ali administrator baze može i eksplicitno odrediti veličinu pojedinih segmenata.

1.1.1. Buffer cache

Kod transformacija (umetanje, brisanje, izmjena) podataka, podaci se ne zapisuju na disk u stvarnom vremenu. Blokovi podataka prvo se pohranjuju u *buffer cache*. Promjene se zapisuju u memoriji te su blokovi neko vrijeme pohranjeni u njoj, sve dok se ne ukaže potreba za dohvaćanjem drugih blokova i oslobađanjem memorijskog prostora. Kod izvršavanja upita, situacije je slična – blokovi podataka prvo se s diska dovlače u *buffer cache,* a potom ih se prebacuje u privatne memorijske PGA segmente (svaka sesija ima zaseban PGA segment). Blokovi podataka su fiksne veličine. Najčešća veličina bloka je 4 KB. Svaka podatkovna datoteka se sastoji od određenog broja blokova podataka na disku. *Buffer cache* se također sastoji od određenog broja blokova podataka u memoriji. Za razliku od blokova podataka koji su fiksne veličine, veličina redaka u tablicama je promjenjiva. Stoga se u jednom bloku može nalaziti više redaka, ali moguće je i da jedan redak zauzima više blokova. Blokovi koji sadržavaju retke kojima se često pristupa trebaju što duže biti u *buffer cache*-u kako bi se izbjeglo često čitanje i pisanje na disk. Blok na disku i blok u *buffer cache*-u su identični u trenutku kada je blok dohvaćen u memoriju. Izmjene nad podacima se zapisuju u memorijskim blokovima. Nakon nekog vremena se blokovi iz memorije zapisuju na disk. Dok se to ne dogodi, podaci u memoriji i na disku su nesinkronizirani. Moguće je da se u nekom bloku izvrše tisuće izmjena prije nego bude zapisan na disk. Veličina *buffer cache*-a je vrlo važna za performanse. Veličina bi trebala biti dostatna za dohvat i pohranu svih blokova kojima se često pristupa, ali opet ne tolika da su u memoriji cijelo vrijeme pohranjeni i blokovi kojima se rijetko pristupa. Ako je ovaj segment premalen, diskovi će cijelo vrijeme biti aktivni, jer će biti potrebno konstantno premještati blokove s diskova u memoriju i obratno. Preveliki *buffer cache* ne mora nužno biti veliki problem, ali ako zauzme previše radne memorije postoji opasnost da će operativni sistem morati koristiti virtualnu memoriju, a to povlači značajnu degradaciju performansi. U verziji *11g* veličina *buffer cache*-a može se dinamički i automatski mijenjati, ali do granica koje odredi administrator baze.

1.1.2. Log buffer

Log buffer je maleni memorijski segment u kojem se pohranjuju transformacijski vektori prije nego ih se zapiše u *redo log* datoteke na disku. Svaka naredba koja mijenja podatke generira

transformacijske vektore. Vektori sadrže informacije o podacima koji su bili zapisani u bloku prije izmjene te o podacima koji su nastali izmjenom. Na ovaj način baza garantira da podaci neće biti izgubljeni u slučaju prekida rada baze, jer se promjene uvijek mogu ponovno izvršiti pomoću transformacijskih vektora. *Log buffer* može u jednom trenutku sadržavati mnoštvo transformacijskih vektora iz različitih transakcija. Pozadinski proces pod nazivom LGWR (*Log writer*) vodi brigu o pohranjivanju *log buffer*-a u memoriju. Svi vektori koji se u nekom trenutku nalaze u *log buffer*-u spremaju se na disk u jednoj *batch* obradi koju pokreće LGWR proces. Međutim, prebacivanje transformacijskih vektora na disk odvija se u gotovo realnom vremenu, a kada korisnik eksplicitno zatraži da njegove izmjene postanu permanentne, obrada se doista izvršava u stvarnom vremenu. Budući da su podaci u *log buffer*-u pohranjeni kroz vrlo kratak vremenski period (jer se pisanje na disk vrši u gotovo realnom vremenu), njegova je veličina malena – svega nekoliko megabajta. Kada korisnik dobije od baze povratnu informaciju da su njegove izmjene nad podacima pohranjene, to znači da je transformacijski vektor zapisan na disk. Proces zapisivanja vektora u *redo log* datoteke predstavlja usko grlo u *Oracle* arhitekturi, jer je nemoguće podatke trajno izmijeniti prije nego LGWR zapiše transformacijske vektore na disk. Veličina *log buffer*-a se ne može mijenjati nakon pokretanja instance.

1.1.3. Shared pool

Shared pool je najkompleksniji SGA segment. Sastoji se od četiri manja segmenta:

- *Library cache* – Sadrži kompajlirani kod SQL naredbi koje su pokretane u prošlosti. Na ovaj se način naredbe mogu ponovno izvršavati bez potrebe za rekompilacijom.
- *Data dictionary cache* – Ovdje su pohranjene definicije objekata iz baze koji su se nedavno koristili – to mogu biti tablice, indeksi, informacije o korisničkim računima i slično. Ovi su objekti dostupni svim sesijama te se smanjuje potreba čitanja rječnika podataka (rječnik podataka sadrži definicije objekata u bazi) s diska.
- *PL/SQL area* – PL/SQL objekti su procedure, funkcije i paketi pohranjeni u bazi, točnije u rječniku podataka. Nakon što sesija prvi put pozove neku proceduru, procedura se sprema u ovaj segment kako bi je naknadni pozivi mogli dohvatiti direktno iz memorije, bez potrebe za čitanjem s diska.
- *SQL query and PL/SQL function result caches* – Čest je slučaj da se identičan upit pokreće više puta. Rezultati takvih upita se spremaju u ovaj segment te se na taj način

sprječava bespotrebno izvršavanje identičnih upita. Mehanizam koji upravlja pohranjivanjem rezultata kontinuirano prati da li je na kojoj od tablica nad kojima se upiti izvršavaju došlo do promjena. Ako je, tada se upit mora ponovno izvršiti, ali ako nije, dovoljno je dohvatiti prethodni rezultat iz ovog spremnika.

Veličina *shared pool*-a je vrlo važna za performanse sustava. Segment bi trebao biti dovoljno velik da može pohraniti sav kod upita i naredbi koje se ponavljaju te definicije objekata koji se u tim naredbama koriste. Međutim, trebalo bi izbjegavati pohranu upita koji se izvršavaju samo jednom. Ako je *shared pool* premalen, sesije će se morati konstantno nadmetati kako bi dobile prostor u kojeg će pohranjivati svoje naredbe i definicije objekata. Kroz kratko vrijeme, taj će prostor biti upotrijebljen za potrebe neke druge sesije te će često čitanje s diskova i pisanje na diskove biti neizbježno. S druge strane, ako je *shared pool* prevelik, potraga za naredbama i definicijama će predugo trajati.

Memorija se unutar *shared pool*-a alocira po LRU (*Least Recently Used*) algoritmu. Kada baza treba prostora za nove upite i definicije u memoriji, prebrisat će one naredbe i definicije koje se najduže nisu koristile. Veličina ovog segmenta može se mijenjati dinamički.

1.2. Fizička struktura podataka

Tri najvažnije skupine datoteka koje čine bazu su:

- Kontrolna datoteka – Sadrži informacije o podatkovnim i *redo log* datotekama. Instanca baze koristi ovu datoteku kako bi pronašla navedene datoteke na disku.
- Podatkovne datoteke – Sadrže poslovne podatke. Baza može sadržavati neograničen broj podatkovnih datoteka
- *Redo log* datoteke – Svaki put kada korisnici žele načiniti neku promjenu nad podacima permanentnom, promjena se prvo zapisuje u *redo log* datoteke, a tek nakon toga u podatkovne datoteke. *Redo log* se koristi u slučaju prekida rada baze kako bi se podaci doveli u prijašnje konzistentno stanje.

1.2.1. Kontrolna datoteka

Kontrolna datoteka najčešće je velika svega nekoliko megabajta, ali je vitalna za rad baze. U njoj su zapisane lokacije podatkovnih datoteka, *redo log* datoteka i arhiviranih *redo log* datoteka (o kojima će biti više riječi u nastavku). Tu su također zapisane informacije o

arhivskim kopijama koje su potrebne u slučaju oporavka baze. Budući da je kontrolna datoteka vrlo važna, uvijek treba biti zaštićena multipleksiranjem – u svakom trenutku trebaju postojati barem 3 identične kopije na različitim diskovima. Kreiranje i održavanje kopija je automatsko, administrator samo treba odrediti broj kopija i njihovu lokaciju. Ako se bilo koja kopija ošteti, bazu neće biti moguće pokrenuti te će se dotična oštećena kopija morati zamijeniti s ispravnom.

1.2.2. *Redo log* datoteke

Redo log datoteke sadrže transformacijske vektore poredane kronološki. Ako se ošteti jedna ili više podatkovnih datoteka, ovi se vektori primjenjuju na arhivske kopije navedenih datoteka kako bi se ponovno izvršile sve transformacije nad podacima i baza vratila u prijašnje stanje. *Oracle* baza pohranjuje *redo log* datoteke u *redo log* grupe. Svaka grupa mora imati barem jednu *redo log* datoteku i moraju postojati barem dvije grupe. Međutim, *redo log* datoteke unutar grupe trebale bi biti zaštićene multipleksiranjem poput kontrolne datoteke. Također, moguće je koristiti i više grupa. Jedna je grupa trenutna i pozadinski proces LGWR (*Log Writer*) zapisuje aktualne transformacijske vektore u tu grupu. *Redo log* datoteke u toj grupi, odnosno članovi te grupe su fiksne veličine. Stoga se s vremenom zapune. Tada druga grupa postaje aktivna, a prva se grupa arhivira. Arhiviranje podrazumijeva kopiranje *redo log* datoteka u arhivske *redo log* datoteke, a pod kontrolom je pozadinskog procesa ARCn. Kada se druga grupa zapuni, opet dolazi do promjene – prva grupa postaje aktivna, a druga se grupa arhivira.

1.2.3. Podatkovne datoteke

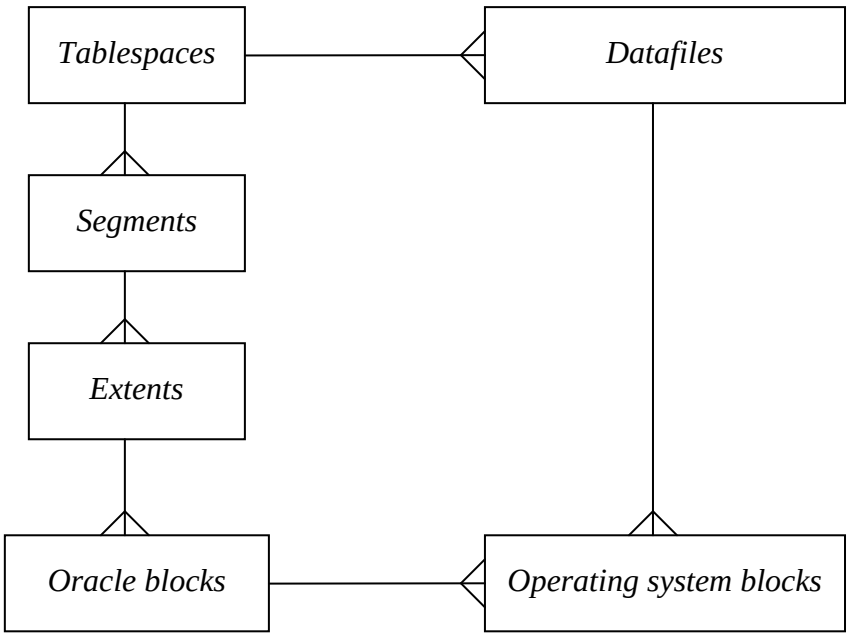
Podatkovne su datoteke repozitorij za podatke. Veličina i broj ovih datoteka nisu ograničeni. Fizički su to datoteke na disku koje su vidljive sistemskim administratorima. Logički, svaka podatkovna datoteka sadrži jedan ili više segmenata (segmenti su tablice i indeksi). Programeri i korisnici pristupaju segmentima i gledaju podatke. Svaka je podatkovna datoteka načinjena od podatkovnih blokova. Blokovi su fiksne veličine koja može varirati od 2KB do 64KB. Blok se sastoji od zaglavlja i prostora za podatke. Blokovi se po potrebi premještaju s diskova u *buffer cache* gdje se vrše transformacije nad podacima. Za razliku od kontrolne datoteke i *redo log* datoteka, baza ne podržava multipleksiranje podatkovnih datoteka. Međutim, one mogu biti multipleksirane nezavisno od baze, npr. uporabom RAID-a. Važno je redovno arhivirati ove datoteke. U slučaju oštećenja, podatkovna datoteka se dohvaća iz

arhive te se na nju primjenjuju svi transformacijski vektori generirani nakon arhiviranja. Treba primijetiti da nije dovoljno primijeniti transformacijske vektore sadržane samo u trenutnoj *redo log* grupi, nego se moraju primijeniti i vektori iz arhiviranih *redo log* grupa kako bi se podaci doveli u konzistentno stanje.

1.3. Logička struktura podataka

Korisnici i programeri vide logičku strukturu podataka, odnosno segmente. Tablice, indeksi i *undo* segmenti (segmenti koji sadrže prijašnje verzije podataka) su najvažniji među njima. Dakle, programeri vide segmente, a sistemski administratori datoteke na disku. Ova apstrakcija logičkog i fizičkog modela ostvarena je pomoću *tablespace*-ova. Logički je *tablespace* kolekcija jednog ili više segmenata, dok se fizički sastoji od jedne ili više podatkovnih datoteka. Baza mora imati minimalno dva *tablespace*-a. To su SYSTEM i SYSAUX *tablespace*. U njima se nalazi rječnik podataka koji sadrži informacije o svim objektima u bazi. Već je rečeno da se segment sastoji od blokova. Kako se segment širi, tako mu se dodjeljuju novi blokovi. Međutim, ne dodaje se jedan po jedan blok, nego se oni grupiraju u *extent*-e. *Extent* se sastoji od više slijedno poredanih blokova unutar podatkovne datoteke. Jedna se tablica sastoji od jednog ili više *extenata* koji pak ne moraju biti poredani jedan iza drugoga, niti moraju biti unutar iste podatkovne datoteke. Slika 1.1 prikazuje odnos između fizičkih i logičkih struktura u bazi.

Slika 1.1 Odnos logičkih i fizičkih struktura u bazi



Veze između podatkovnih datoteka i *tablespace*-ova su pohranjene u kontrolnoj datoteci – za svaku podatkovnu datoteku postoji zapis koji govori kojem *tablespace*-u pripada. Nemoguće je pokrenuti instancu baze bez korištenja kontrolne datoteke, jer su u njoj između ostalog pohranjene lokacije podatkovnih datoteka koje čine SYSTEM i SYSAUX *tablespace*-ove. U ta se dva *tablespace*-a nalaze tablice koje čine rječnik podataka, a u njemu su pohranjene informacije o svim objektima u bazi.

2. Administracija korisnika

Korisnici se spajaju na bazu pomoću korisničkih računa. Za svaki su korisnički račun definirane privilegije i prava nad objektima u bazi. Budući da postoji mnogo različitih privilegija, nepraktično ih je zasebno dodjeljivati korisnicima. Stoga se privilegije mogu grupirati u role, a role se mogu dodijeliti korisniku. *Oracle* baza nudi i opciju ograničavanja resursa baze koje pojedini korisnici mogu koristiti. Ta je kontrola implementirana uz pomoć profila – za svaki se korisnički račun može definirati zasebni profil. Profili omogućuju i administraciju zaporki.

2.1. Dodjeljivanje prava korisnicima

Svakom se korisničkom računu moraju dodijeliti određene privilegije kako bi korisnik mogao koristiti bazu. Sistemske privilegije omogućuju korisnicima spajanje na bazu, kreiranje objekata u bazi, dodavanje novih korisnika i slično. Prava nad objektima pak omogućuju izvođenje upita i mijenjanje podataka u tablicama. Korisniku se može omogućiti da sistemske privilegije i prava nad objektima prenosi drugim korisnicima. *Oracle* baza funkcionira na *least privilege* principu. To znači da bi svaki korisnik trebao imati samo minimum privilegija koje mu trebaju da obavlja posao. Stoga novi korisnici ne mogu raditi apsolutno ništa – ne mogu se čak niti spojiti na bazu dok im se ne dodijeli sistemska CREATE SESSION privilegija. Ako se tada novi korisnik spoji na bazu, opet mu to neće mnogo značiti, jer nema pravo na pregled ili kreiranje niti jedne tablice – administrator mu mora dodijeliti SELECT TABLE i CREATE TABLE privilegije. Korisnicima se može omogućiti da svoja prava prenose drugim korisnicima.

2.2. Role

Ako se u bazi podataka nalazi nekoliko tisuća tablica i ako je broj korisnika velik, dodjeljivanje pojedinačnih privilegija svakom korisniku je mukotrpan i dugotrajan posao za administratore. Također, kada se privilegija dodijeli korisniku, on je može koristiti u bilo kojem trenutku, a to nekad nije poželjno (na primjer, ako administrator želi zabraniti pristup tablicama u vrijeme dok se na bazi analizira njihova veličina i struktura indeksa kako bi se

upiti automatski mogli optimizirati). Oba navedena problema se mogu riješiti primjenom rola. Rola je skup sistemskih privilegija i privilegija nad objektima u bazi. Ta se prava dodjeljuju skupno te se mogu aktivirati ili deaktivirati unutar sesije.

Na primjer, neka postoje tri skupine korisnika: pripravnici, zaposlenici i menadžeri. Menadžeri mogu pregledavati i mijenjati podatke u svim tablicama, zaposlenici mogu pregledavati podatke u svim tablicama, ali mijenjati samo neke, dok pripravnici ne mogu mijenjati podatke niti u jednoj tablici. Privilegije se mogu rasporediti u tri role:

```
create role hr_junior;
grant create session to hr_junior;
grant select on hr.regions to hr_junior;
grant select on hr.locations to hr_junior;
grant select on hr.countries to hr_junior;
grant select on hr.departments to hr_junior;
grant select on hr.job_history to hr_junior;
grant select on hr.jobs to hr_junior;
grant select on hr.employees to hr_junior;
```

Korisnici kojima je dodijeljena rola *hr_junior* mogu se spojiti na bazu i dohvatiti podatke iz navedenih tablica.

```
create role hr_senior;
grant hr_junior to hr_senior with admin option;
grant insert, update, delete on hr.employees to hr_senior;
grant insert, update, delete on hr.job_history to hr_senior;
```

Hr_senior rola omogućuje pregled svih tablica, ali i izmjenu podataka u *employees* i *job_history* tablicama.

```
create role hr_manager;
grant hr_senior to hr_manager with admin option;
grant all on hr.regions to hr_manager;
grant all on hr.locations to hr_manager;
grant all on hr.countries to hr_manager;
grant all on hr.departments to hr_manager;
grant all on hr.job_history to hr_manager;
grant all on hr.jobs to hr_manager;
grant all on hr.employees to hr_manager;
```

Hr_manager rola ima prava za pregled i izmjenu podataka u svim tablicama. Treba primijetiti

da je CREATE SESSION jedina sistemska privilegija dodijeljena *hr_manager* roli i to preko *hr_senior* role. Dakle, menadžeri ne mogu niti kreirati niti brisati tablice.

Ako je korisniku dodijeljena rola, to ne mora značiti da je ona i aktivna cijelo vrijeme dok je korisnik spojen na bazu. Rola se može kreirati uz pomoć sintakse:

```
CREATE ROLE rolename IDENTIFIED USING procedure_name;
```

Na ovaj se način rola može aktivirati samo pokretanjem procedure koja može napravit niz provjera – sa koje se IP adrese korisnik spojio na bazu, koje je doba dana, koju aplikaciju koristi i tako dalje. Ako neki od uvjeta nije ispunjen, procedura javlja grešku i rola neće biti aktivirana, iako je korisnik spojen na bazu.

2.3. Profili

Profili omogućuju implementaciju pravila za korištenje zaporki i ograničavanje resursa korisnicima. Kontrola resursa ne mora biti aktivna cijelo vrijeme, nego je administrator može aktivirati po potrebi na razini cijele baze. Svakom je korisniku inicijalno dodijeljen profil DEFAULT.

Važniji parametri profila za upravljanje lozinkama su:

- FAILED_LOGIN_ATTEMTPS – Broj neuspjelih uzastopnih pokušaja prijave na bazu nakon kojeg se zaključava korisnički račun.
- PASSWORD_LOCK_TIME – Definira koliko će dana korisnički račun biti zaključan nakon krivog unosa lozinke *n* puta.
- PASSWORD_LIFE_TIME – Definira nakon koliko se dana lozinka mora promijeniti.
- PASSWORD_VERIFY_FUNCTION – Svaki put kada se lozinka promijeni, pokreće se funkcija navedena u ovom parametru kako bi provjerila da li nova lozinka zadovoljava željene kriterije.

Neki od parametara kojima se ograničavaju resursi su:

- SESSIONS_PER_USER – Broj sesija koje korisnik može istovremeno pokrenuti.
- CPU_PER_SESSION – Definira koliko procesorskog vremena jedna sesija maksimalno smije iskoristiti.
- PRIVATE_SGA – Definira koliko kilobajta jedna sesija smije zauzeti u SGA

memorijskom segmentu.

- CONNECT_TIME – Maksimalno dozvoljeno trajanje sesije.

Na primjer, neka je politika firme da samo administratori mogu uspostaviti neograničen broj sesija na bazu, programeri u jednom trenutku mogu koristiti maksimalno dvije sesije, a korisnici samo jednu. Administratori moraju mijenjati zaporku jednom tjedno. Navedeni se zahtjevi mogu ispuniti na sljedeći način:

```
alter profile default limit sessions_per_user 1;
create profile dba_profile limit sessions_per_user unlimited
password_life_time 7 password_grace_time 1;
alter user sys profile dba_profile;
create profile programers_profile limit_sessions_per_user 2;
alter user scott profile programmers_profile;
```

3. Konkurentno izvođenje transakcija

U nekom trenutku bazu mogu koristiti stotine korisnika. Postoji vjerojatnost da će više korisnika istovremeno pristupati istim podacima i pokušavati ih modificirati. Svaka baza podataka mora podržavati ACID (*Atomicity, Consistency, Isolation, Durability*) koncept. *Oracle* koristi mehanizme zaključavanja tablica i redaka za vrijeme izvođenja transakcija, podatke iz *undo* segmenata i *redo log* datoteke kako bi osigurao atomarnost, konzistentnost, izolaciju i trajnost podataka u bazi.

3.1. ACID koncept

Atomarnost podrazumijeva da se svaka naredba unutar transakcije mora uspješno izvršiti. Ako to nije moguće, onda se moraju poništiti efekti cijele transakcije, a ne samo onaj dio koji je uzrokovao grešku. Na primjer, ako se u tablicu rezervacija avionskih karata unese novi zapis, nužno je smanjiti broj slobodnih mjesta u avionu za jedan. Obje se naredbe moraju uspješno izvršiti kako bi podaci u bazi bili ispravni.

Rezultat upita mora biti **konzistentan** sa stanjem podataka u bazi na početku izvođenja transakcije. Upiti nad velikim tablicama mogu trajati minutama pa i satima. Ako je neki drugi korisnik za vrijeme trajanje upita promijenio podatke u dotičnoj tablici, korisnik koji je pokrenuo upit toga ne smije biti svjestan.

Pojam **izolacije** podrazumijeva da se izmjene nad podacima mogu vidjeti tek nakon što transakcija završi i nakon što korisnik koji je pokrenuo transakciju potvrdi promjene (naredba COMMIT).

Nakon što korisnik potvrdi transakciju, izmjene nad podacima moraju zauvijek ostati u bazi, odnosno baza mora podržavati **trajnost** podataka.

3.2. Implementacija ACID koncepta

Upiti dohvaćaju podatke za pregled. Ako se za vrijeme izvršavanja upita dogode promjene nad tablicama, *Oracle* baza mora osigurati da se te promjene ne vide u rezultatu upita. Ovo se postiže uporabom *undo* segmenata. Naime, dok god traje izmjena nad podacima, stara se

verzija podataka čuva u *undo* segmentima kako bi mogla opslužiti upite koji su startali prije nego je izmjena počela. Upit može započeti i nakon što je transakcija završila, ali prije nego je potvrđena (COMMIT) od korisnika koji je izvršio transakciju. Baza u tom slučaju također koristi *undo* segmente kako bi podržala svojstvo izolacije.

Naredbe kojima se izmjenjuju podaci koriste *undo* segmente i transformacijske vektore. Transakcija zaključava sve retke koje treba izmijeniti i ne dozvoljava drugim transakcijama bilo kakve izmjene. Nakon toga se u transformacijske vektore zapisuju sve promjene koje će se izvršiti. Nove se vrijednosti zapisuju u podatkovne blokove, a stare se vrijednosti zapisuju u *undo* blokove. Svi upiti koji koriste modificiranu tablicu usmjeravaju se na *undo* blokove sve do trenutka kada korisnik potvrdi izmjene. Jedino sesija koja vrši izmjene može vidjeti aktualne promijenjene podatke u periodu kada su izmjene načinjene, ali ne i potvrđene.

U slučaju kada nepotvrđenu transakciju treba poništiti, opet se koriste *undo* segmenti i transformacijski vektori. U slučaju UPDATE naredbe, stara verzija podataka iz *undo* segmenata se koristi kako bi se generirala naredba koja trenutnu verziju podataka vraća u prijašnje stanje. INSERT naredbu je vrlo jednostavno poništiti – baza dohvaća identifikatore umetnutih redaka (ROWID) iz *undo* segmenta te konstruira DELETE naredbu kojom se podaci u memoriji vraćaju u prvobitno stanje. Kod poništavanja efekata DELETE naredbe situacija je obratna – podaci iz *undo* blokova se koriste za generiranje INSERT naredbe koja će poništiti brisanje.

Podaci nad kojima se vrše izmjene nalaze se u memoriji. Kada korisnik potvrdi izmjene, moraju li se ti podaci zapisati u podatkovne datoteke na disku? Ne moraju, jer se u transformacijskim vektorima nalaze sve potrebne informacije koje omogućavaju ponovno kreiranje i izvršavanje upita koji će bazu dovesti u konzistentno stanje. To znači da se kod potvrde izmjena na diskove moraju zapisati samo transformacijsku vektori. Na primjer, ako se baza ugasi zbog prekida napajanja, ona se može dovesti u stanje u kojem je bila primjenom transformacijskih vektora na arhivirane podatkovne datoteke. Niti jedna potvrđena transakcija neće biti izgubljena. Međutim, efekti nepotvrđenih transakcija i transakcija koje su se izvršavale u trenutku iskapčanja bit će izgubljeni.

4. Izrada sigurnosnih kopija i oporavak baze

Postoje različiti tipovi grešaka i problema koji se mogu javiti u radu s bazom podataka. Oporavak od nekih grešaka je automatski, dok druge zahtijevaju intervenciju administratora. Ako SQL naredba uzrokuje grešku, baza automatski poništava efekte naredbe korištenjem *undo* podataka i transformacijskih vektora. Pogreška se može dogoditi i na razini korisničkog procesa koji je pokrenuo transakciju. Najčešći je uzrok prekid veze između klijenta i poslužitelja. Ovakve slučajeve rješava pozadinski proces PMON na način da poništava efekte transakcije i gasi serverski proces te oslobađa zauzetu memoriju. Tipičan problem koji svakako zahtijeva angažman administratora je oštećenje diskova koji čuvaju podatke. Kontrolna datoteka i *redo log* datoteke moraju u svakom trenutku biti zaštićene multipleksiranjem. Ako se dotične datoteke multipleksiraju na različite diskove, uvijek je moguće upotrijebiti kopije s ispravnih diskova. Budući da postoji malena vjerojatnost da se pokvare svi diskovi na kojima se nalaze multipleksirane kopije, dodatne je mjera sigurnosti arhiviranje kontrolne i *redo* log datoteka. Podatkovne datoteke se ne mogu multipleksirati, stoga je jedina opcija kreiranje sigurnosnih arhivskih kopija. Kako bi se podaci doveli u konzistentno stanje, na arhivske je kopije potrebno primijeniti transformacijske vektore iz arhiviranih i aktualnih *redo log* datoteka. Koriste se oni transformacijski vektori u kojima su zapisane promjene koje su se dogodile u periodu između trenutka kreiranja arhivske kopije podatkovne datoteke i kvara na mediju.

4.1. Prekid rada instance

Tipičan uzrok prekida rada instance je nestanak struje. Ako je do gašenja došlo u trenutku zapisivanja promjena u podatkovne datoteke, podaci nisu konzistentni. Naime, zapisivanje promjena u podatkovne datoteke ne obavlja se sinkrono s potvrđivanjem promjena od strane korisnika (COMMIT naredba). Međutim, transformacije nad podacima se zapisuju u *redo log* datoteke u trenutku kada korisnik potvrdi promjene. Stoga su informacije u *redo log* datotekama uvijek konzistentne sa stvarnim stanjem podataka. Nakon ponovnog pokretanja instance, oporavak je u potpunosti automatski i sastoji se od primjene transformacijskih vektora na nekonzistentne podatke u podatkovnim datotekama na disku. Nakon dohvata podataka s diska i izvršavanja naredbi zapisanih u *redo log* transformacijskim vektorima,

podaci opet postaju konzistentni. U poglavlju 1.2.2. objašnjeno je da se *redo log* datoteke nalaze u grupama. Nakon što se datoteka zapuni, trenutna grupa postaje neaktivna, a sljedeća aktivna. Kod oporavka baze u slučaju iznenadnog gašenja instance koriste se samo aktivne *redo log* datoteke, dok se arhivirane ne koriste. Stoga baza zapisuje sve promjene iz *redo log* grupe u podatkovne datoteke prije nego sljedeća grupa postane aktivna. Administrator mora voditi računa i o trajanju operacije oporavka, odnosno trajanju ponavljanja izmjena nad podacima. Podešavanjem parametara u bazi se može eksplicitno definirati maksimalno dopušteno trajanje oporavka. To će rezultirati češćim zapisivanjem izmjena u podatkovne datoteke i manjom trenutnom količinom transformacijskih vektora u *redo log* datotekama. Stoga je u slučaju oporavka potrebno ponoviti manji broj izmjena nad podacima pa je i vrijeme oporavka kraće. Međutim, treba biti vrlo oprezan kod definiranja željenog trajanja oporavka. Što je oporavak kraći, zapisivanje podataka na diskove tijekom normalnog rada je intenzivnije, a to može znatno reducirati performanse sustava.

4.2. Kvar medija za pohranu podataka

Kao što je rečeno u prethodnom poglavlju, oporavak od iznenadnog gašenja instance je automatski i sastoji se od primjene transformacija u *redo log* datotekama na podatke zapisane na disku. U tom slučaju upotreba arhivskih *redo log* datoteka nije potrebna. Međutim, oporavak od kvara na diskovima koji sadrže podatkovne datoteke je složeniji. Budući da se *redo log* datoteke nakon nekog vremena izbrišu i ponovno pune novim transformacijskim vektorima, prijašnje promjene su izgubljene. Stoga je potrebno arhivirati *redo log* datoteke nakon svake promjene aktivne *redo log* grupe (poglavlje 1.2.2.). Rezultat toga je postojanje niza arhivskih *redo log* datoteka u kojima su zapisane sve promjene nad podacima ikad načinjene. Ako se podatkovna datoteka ošteti, oporavak se sastoji od sljedećih koraka:

- Dohvat najsvježije arhivske kopije podatkovne datoteke.
- Primjena transformacijskih vektora iz arhivskih *redo log* datoteka.
- Primjena transformacijskih vektora iz aktivne *redo log* datoteke.

Oracle baza je inicijalno u NOARCHIVELOG načinu rada. To znači da se transformacijski vektori ne arhiviraju i podaci su nezaštićeni u slučaju kvara medija. Iskusan će administrator stoga odmah nakon instalacije bazu prebaciti u ARCHIVELOG način rada. To rezultira pokretanjem pozadinskog procesa koji brine o arhiviranju *redo log* datoteka nakon svake izmjene aktivne *redo log* grupe. Lokacija za arhiviranje se eksplicitno određuje. Dodatna je

zaštita multipleksiranje arhiviranih *redo log* datoteke na različite diskove. Trajanje oporavka ovisi o frekvenciji arhiviranja podatkovnih datoteka. Što je arhiva svježija, potrebno je primijeniti manji broj transformacijskih vektora na podatke pa je oporavak kraći. Međutim, arhiviranje podataka usporava bazu. Stoga administrator mora odabrati optimalno rješenje čijom primjenom baza neće biti preopterećena arhiviranjem dok će potencijalni oporavak od kvara medija biti izvršen u razumnom vremenu.

5. Zaključak

Baze podataka sadrže važne informacije o poslovanju. Podaci u njima predstavljaju sliku trenutnog stanja firme i poslovnih procesa. Povijesni pregled podataka predstavlja temelj za ekstrakciju znanja i potporu poslovnom odlučivanju. Stoga svaka ozbiljna baza mora zadovoljiti određene kriterije. Ti se kriteriji odnose na kontrolu pristupa podacima, izvođenje konkurentnih upita i transformacija nad podacima, podržavanje ACID koncepta te zaštitu od gubitka podataka. U sklopu ovog rada prezentirana su rješenja navedenih zahtjeva implementirana u bazi podataka *Oracle Database 11g*.

Memorijske strukture i pozadinski procesi tvore instancu baze. Podatkovne strukture se mogu promatrati s dva aspekta – logičkog i fizičkog. Logički su podaci pohranjeni u segmentima (tablice, indeksi), dok su fizički pohranjeni u podatkovnim datotekama. Osim podatkovnih, u bazi su prisutne kontrolna datoteka te *redo log* datoteke. Kontrolnu datoteku koristi instanca prilikom pokretanja dok se u *redo log* datoteke pohranjuju transformacijski vektori.

Različiti korisnici posjeduju različite privilegije pregleda i izmjene podataka. Ove se privilegije mogu dodjeljivati pojedinačnim korisnicima ili grupama korisnika upotrebom rola. Restrikcija resursa se obavlja definiranjem i dodjeljivanjem profila.

Undo segmenti, *redo log* datoteke te mehanizmi zaključavanja redaka i tablica podupiru izvođenje konkurentnih transakcija i implementaciju ACID koncepta. Uporabom navedenih tehnika baza prelazi iz jednog konzistentnog stanja u drugo što automatski povlači i konzistentnost podatka u svakom trenutku.

Gubitak podataka uslijed bilo kakvog kvara je nedopustiv u korporativnom okruženju. U slučaju prekida rada instance, oporavak je potpuno automatski. Pozadinski procesi tijekom ponovnog pokretanja koriste podatkovne datoteke i transformacijske vektore kako bi ponovno izvršili potvrđene izmjene nad podacima koje u trenutku iskapčanja nisu bile zapisane na diskove. S druge strane, oporavak od kvara medija za pohranu podataka nije automatiziran i zahtijeva akciju administratora. Osim arhivske kopije podataka, u ovom se slučaju moraju koristiti i arhivirani i trenutni transformacijski vektori kako bi se podaci vratili u ispravno stanje. Što se podatkovne datoteke češće arhiviraju, to je oporavak kraći, jer je potrebno ponoviti manji broj izmjena nad podacima. Dodatnu mjeru zaštite kontrolne i *redo log* datoteka predstavlja multipleksiranje na različite diskove.

Literatura

[1] WATSON J. *Oracle Database 11g: Administration I*

[2] http://www.infoprofil.info/edu/baze_edu.htm

[3] http://www.oracle.com/pls/db111/homepage

[4] http://databases.about.com

[5] http://www.dba-oracle.com