Univerzitet u Nišu
Elektronski fakultet
Katedra za
računarstvo

SISTEMI ZA UPRAVLJANJE BAZAMA PODATAKA

Obrada transakcija, planovi izvršavanja transakcija, izolacija i zaključavanje kod Oracle DBMS-a

Profesor:
Prof. dr Aleksandar Stanimirović

Student:
Bojana Svilenković 1417

Niš, 2022.

Sadržaj

1. Uvod

Da bi sistem za upravljanje bazom podataka funkcionisao na prihvatljiv i ispravan način potrebno je da zna kako treba da radi u višekorisničkom okruženju, gde se upiti izvršavaju istovremeno, odnosno u kome je prisutna konkurentnost. Tu se pre svega mora voditi računa o tome u kakvom je stanju baza u svakom trenutku, tj stanje mora biti konzistentno. Ne sme se ugroziti ispravnost podataka ni u kom trenutku, što znači da se moraju zabraniti polovične, odnosno nepotpuno uspešne operacije, kao i one koje bi ugrozile tačnost ostalih operacija koje su u toku. Sa druge strane, tu su vremeneska ograničenja koja, zbog zahteva o konzistentosti, nije lako ispuniti. Zbog svega ovoga koncept transakcija je jako bitan u bilo kom sistemu za upravljanje bazama podataka. One obezbeđuju da se skup operacija izvršava kao celina, po principu sve ili ništa, čime direktno utiče na to da stanje u bazi bude konzistentno i da se tamo nađu samo ispravni podaci.

2. Transakcija

Baza podataka je zajednički resurs koji istovremeno (konkuretno) korisi veći broj programa. Pri ovakvom korišćenju baze podataka može doći do mnogih neželjenih efekata kao što su, na primer:

- otkaz sistema u toku izvršenja nekog programa koji može da ostavi bazu podataka u nekonzistentnom stanju,
- neželjena interferencija dva ili više programa nad podacima za koje istovremeno konkurišu, može, takođe, da dovede bazu podataka u nekonzistentno stanje.

U cilju rešavanja ovih problema baze podataka podržavaju korišćenje transakcija da bi se obezbedilo konkurentno izvođenje akcija, ali i u cilju oporavka baze podataka.

Transakcija se definiše kao jedinica posla, koju čine jedna ili više SQL naredbi koje izvršavaju odgovarajući skup akcija. Transakcija grupiše SQL naredbe tako da su ili sve commit-ovane, što znači da su sve primenjene na bazu podataka ili su sve rollback-ovane, što znači da su efekti svih naredbi u transakcij poništeni.

U Oracle-u, sve transakcije moraju da poštuju osnovne principe transakcija, poznate kao ACID svojstva. ACID predstavlja skup svojstava transakcije u bazama podataka namenjenih da garantuju validnost čak i u slučaju grešaka, nestanka struje, itd. Četiri osobine koje su koncept ACID svojstava opisuju glavne garancije za paradigmu transakcije koja je uticala na mnoge aspekte razvoja u sistemima baza podataka. ACID je akronim za sledeće:

- ***A**tomicity – Atomičnost*

  Sve naredbe jedne transakcije su izvršene ili nije nijedna, tj. transakcija uspeva potpuna ili ne uspeva u potpunosti. Ne postoje parcijalne transakcije. Na primer, ako transakcija počne da ažurira 100 reda jedne tabele, ali iz nekog razloga dođe do greške nakon 20 ažuriranja, onda će baza podataka rollback-ovati, poništiti, promene koje su primenjene na tih 20 redova.

- ***C**onsistency – Konzistentnost*

  Konzistentnost osigurava da transakcija može prevesti bazu podataka samo iz jednog važećeg stanja u drugo, održavajući invarijantne baze podataka: svi podaci upisani u bazu podataka moraju biti važeći u skladu sa svim definisanim pravilima, uključujući ograničenja, kaskade, okidače i svaku njihovu kombinaciju. Ovo sprečava korupciju baze podataka nezakonitim transakcijama, ali ne garantuje da je transakcija tačna. Na primer, u bankovnom sistemu, neuspela transakcija koja prebacuje novac sa jednog računa na drugi ne sme dovesti do stanja baze podataka takvo da je novac skinut sa jednog računa a nije dodat na drugi račun.

- ***I**solation – Izolacija*

  Transakcije se često izvršavaju istovremeno (npr. čitanje i upis u više u tabela istovremeno) . Izolovanost osigurava da istovremeno izvršenje transakcija ostavja bazu podataka u istom stanju koje bi se dobilo ukoliko su transakcije izvršene sekvencijalno. Izolovanost je glavni cilj kontrole konkurencije. U zavisnosti od korišćenog metoda, efekti nepotpune transakcije možda neće biti vidljivi ni za druge transakcije, što je slučaj kod Oracle baze podataka.

- ***D**urability – Izdržljivost*

  Izdržljivost garantuje da će, nakon što se transakcija završi, ostati sigurna čak i u slučaju otkazivanja sistema (npr. nestanka struje ili pada sistema). Ovo obično znači da su izvršene transakcije (ili njihovi elementi) zapisani u nepromenljivoj memoriji. Promene commit-ovane transakcije su permanentne.

Korišćenje transakcija je jedan od najvažnijih načina na koji se sistemi baza podataka razlikuju od fajl sistema.

3. Obrada transakcija

3.1 Početak transakcije

Transakcija počinje onda kada se naidje na prvu izvršivu SQL naredbu. Izvršiva SQL naredba je SQL naredba koja generiše pozive ka instanci baze podataka, uključujući DML i DDL izraze, kao i *SET TRANSACTION* naredbu.

Kada transakcija počne, Oracle baza podataka dodeljuje transakciju dostupnom segmentu podataka za poništavanje, *undo data segment,* da bi se zabeležili unosi za opoziv transakcije. ID transakcije se ne dodeljuje dok se ne dodele segmenti za poništavanje i slot tabele transakcija, što se dešava tokom prve DML naredbe. ID transakcije je jedinstven za transakciju i predstavlja broj segmenta za poništavanje, slot i redni broj.

```
SQL> UPDATE hr.employees SET salary=salary;

107 rows updated.

SQL> SELECT XID AS "txn id", XIDUSN AS "undo seg", XIDSLOT AS "slot",
  2  XIDSQN AS "seq", STATUS AS "txn status"
  3  FROM V$TRANSACTION;

txn id             undo seg       slot        seq txn status
---------------- ---------- ---------- ---------- ----------------
0600060037000000          6          6         55 ACTIVE
```

Na slici je prikazan primer početka transakcije. Nad tabelom *Employees* se vrši ažuriranje podataka. Transakcija počinje prvom naredbom. Vidimo da je 107 reda tabele ažurirano, međutim, kako transakcija nije potvrđena, tj. nije commit-ovana, te promene još uvek nisu vidljive. SELECT naredba prikazuje detalje o transakciji koju smo započeli. Vidmo slot i segment za poništavanje koji su dodeljeni našoj transakciji, kao i ID transakcije i njen trenutni status. Transakcija je u aktivnom statusu jer još uvek nije okončana.

3.2 Kraj transakcije

Transakcija se može okončati pod različitim uslovima. Kraj transakcije nastaje kada dođe do jedne od sledećih akcija:

- Korisnik izda *COMMIT* ili *ROLLBACK* naredbu bez *SAVEPOINT*-a

  Koristeći *COMMIT,* korisnik eksplicitno navodi zahtev da promene koje su nastale transakcijom budu permanentno upisane u bazu podataka. Promene su od tog trenutka vidljive drugim korisnicima.

  Suprotno od toga, ako korisnik izda *ROLLBACK* naredbu, promene koje su nastale zbog transakcije se brišu i baza podataka ostaje nepromenjena.

- Korisnik pokrene DDL komandu (npr. *CREATE, DROP, RENAME, ALTER, ...)*

  Oracle baza podataka izdaje implicitnu *COMMIT* naredbu pre i posle svake DDL naredbe. Ako trenutna transakcija sadrži DML naredbe, onda Oracle baza podataka najpre vrši commit-ovanje transakcije pa tek onda izvršava DDL naredbu kao novu transakciju sa jednom naredbom.

- Korisnik na korektan način izađe iz većine Oracle Database alata, što izaziva da se trenutna transakcija implicitno commit-uje. Osobina aplikacije da izvrši implicitni commit kada se korisnik diskonektuje od baze je podesiva osobina drajvera.

- Klijentski proces se prekine abnormalno pri tom izazivajući da se transakcija rollback-uje korišćenjem podataka sačuavnih u tabeli transakcija i segmentu za poništavanje.

Nakon što se završi jedna transakcija, sledeća izvršna SQL naredba automatski započinje novu transakciju. Sledeći primer ilustruje ažuriranje, koje započinje transakciju, završavanje transakcije *ROLLBACK* naredbom, a potom ponovo ažuriranje koje započinje novu transakciju, što se primećuje upoređivanjem identifikatora transakcije (XID):

```
SQL> UPDATE hr.employees SET salary=salary;
107 rows updated.

SQL> SELECT XID, STATUS FROM V$TRANSACTION;

XID              STATUS
---------------- ----------------
0800090033000000 ACTIVE

SQL> ROLLBACK;

Rollback complete.

SQL> SELECT XID FROM V$TRANSACTION;

no rows selected

SQL> UPDATE hr.employees SET last_name=last_name;

107 rows updated.

SQL> SELECT XID, STATUS FROM V$TRANSACTION;

XID              STATUS
---------------- ----------------
0900050033000000 ACTIVE
```

3.3 *SAVEPOINT* u transakcijama

U Oracle bazi podataka, postoji mogućnost deklarisanja markera u okviru transakcije koji omogućava parcijalni rollback. Dok se transakcija izvršava moguće je kreiranje *SAVEPOINT-a* kako bi se označile različite tačke unutar transakcije. Ako tokom obrade transakcije dođe do neke greške, možemo izvršiti rollback do željenog *SAVEPOINT*-a ili skroz do početka transakcije.

Kada se transakcija rollback-uje do *SAVEPOINT*-a, dogodi se sledeće:

1. Oracle baza podataka rollback-uje samo one naredbe koje su se izvršile nakon *SAVEPOINT*-a,

2. Oracle baza podataka čuva *SAVEPOINT* do kog je urađen rollback, ali su svi *SAVEPOINT*-i koji su deklarisani nakon njega “izgubljeni”,

3. Oracle baza podataka otključava sve tabele i redove koje je zaključala nakon tog *SAVEPOINT*-a ali zadržava zaključanim sve što je u transakciji zaključano pre njega.

```
SQL> INSERT INTO AUTHOR
2 VALUES ('A111', 'john',
3 'garmany', '123-345-4567',
4 '1234 here st', 'denver',
5 'CO','90204', '9999');

1 row created.

SQL> savepoint in_author;

Savepoint created.

SQL> INSERT INTO BOOK_AUTHOR VALUES ('A111', 'B130', .20);
1 row created.

SQL> savepoint in_book_author;

Savepoint created.

SQL> INSERT INTO BOOK
2 VALUES ('B130', 'P002', 'easy oracle sql',
3 'miscellaneous', 9.95, 1000, 15, 0, '',
4 to_date ('02-20-2005','MM-DD-YYYY'));
1 row created.

SQL> rollback to in_author;

Rollback complete.
```

U primeru sa slike je započeta transakcija DML naredbom za upis podataka u tabelu *AUTHOR*. Nakon upisa podataka, u okviru transakcije je kreiran *SAVEPOINT in_author*. Nakon *SAVEPOINT-a* unete su vrednosti u tabelu *BOOK_AUTHOR* i napravljen je još jedan *SAVEPOINT* sa nazivom *in_book_author.* Na kraju je u tabelu *BOOK* unet red i izvršen je rollback do *SAVEPOINT-a in_author,* što je poništilo unos podataka u sve tabele osim prve.

3.4 Commit mehanizam sa dve faze

U distribuiranoj bazi podataka, Oracle mora da koordiniše obradom transakcije preko mreže i održi konzistenciju podataka, čak i ako dođe do otkaza na mreži ili u sistemu.

Distribuirana transakcija je transakcija koja uključuje jednu ili više naredbi koje ažuriraju podatke na dve ili više instance distribuirane baze podataka.

Commit mehanizam sa dve faze garantuje da će sve instance baza podataka koje učestvuju u distribuiranoj transakciji commit-ovati ili da će sve rollback-ovati promene nastale u toku transakcije. Ovaj mehanizam je potpuno transparentan korisniku koji koristi distribuiranu transakciju. Šta više, korisnik nije ni svestan da je sama transakcija distribuirana. COMMIT naredba koja označava kraj transakcije automatski pokreće mehanizam sa dve faze.

4. Planiranje transakcija

Oracle DBMS vidi transakcije kao seriju, ili spisak (listu), akcija. Akcije koje mogu da se izvrše transakcijama uključuju čitanja i upisivanja objekata baze podataka. Transakcije, takođe, mogu da se definišu kao skup akcija koje su parcijalno uređene tj. relativni redosled nekih akcija ne mora da bude značajan.

Zbog jednostavnosti zapisa i razmaztranja pojedinih slučajeva, uvodimo sledeće pretopstavke i oznake:

- Podrazumeva se das u objekti O nad kojima se izvršavaju operacije uvek učitani u promenljivu koje se takođe naziva O,
- RT(O) predstavlja akciju transakcije T koja čita objekat O,
- WT(O) predstavlja akciju transakcije T koja upisuje objekat O,
- Kada je T izbrisana iz konteksta biće izostavljen indeks.

Vremensko planiranje je spisak akcija (čitanja, upisivanja, prekidanja ili potvrđivanja) iz skupa transakcija, a uređenje u kome se dve akcije transakcije T pojavljuju u vremenskoj raspodeli mora da bude isto kao uređenje u kome se one pojavljuju u T. Vremensko planiranje predstavlja neku stvarnu ili potencijalnu izvođačku sekvencu. Na primer, vremensko planiranje u donjoj tabeli prikazuje uređenje izvođenja dve transakcije *T1* i *T2*. Izvršavanje počinje od prve vrste. Vremensko planiranje opisuje akcije transakcije kako ih vidi DBMS.

T1	T2
R(A)	
W(A)	
	R(B)
	W(B)
R(C)	
W(C)	

Vremensko planiranje u ovoj table ne sadrži prekid ili potvrđivanje akcija za bilo koju transakciju. Vremensko planiranje koje sadrži bilo prekid ili potvrđivanje za svaku transakciju, čije su akcije nabrojane u njoj, naziva se kompletno vremensko planiranje. Kompletno vremensko planiranje mora da sadrži sve akcije svih transakcije koje se pojavljuju u njemu. Ako se akcije različitih transakcija ne prepliću tj. transakcije se izvršavaju od početka do kraja jedna po jedna, takvo vremensko planiranje naziva se serijsko vremnesko planiranje.

4.1 Konkurentno izvršavanje transakcija

Oracle DBMS prepliće akcije različitih transakcija da bi povećao performanse, tako što povećava propusnu moć ili povećava vreme odziva za kratke transakcije. Dok jedna transakcija čeka na stranicu koja će biti pročitana sa diska, CPU može da obradi drugu transakciju. Preklapanje aktivnosti I/O i CPU povećava propusnu moć sistema. Preplitanje izvršavanja kratkih transakcija sa dugim transakcijama obično omogućava kratkim transakcijama da se brže kompletiraju. U serijskom izvršavanju, kratke transakcije bi mogle da se nađu iza dugih transakcija uvodeći nepredvidivo odlaganje u vremenu odziva ili prosečnog vremena koje je uzeto za kompletiranje transakcija.

Konkurento izvršavanje transakcija može dovesti do nekih anomalija koje mogu ostaviti bazu podataka u nekonzistentnom stanju. Dve akcije nad istim objektom podataka biće u konfliktu ako je najmanje jedna od njih upisana. Kada imamo dve potvrđene (commit-ovane) transakcije, koje bi mogle da se izvedu nad konzistentnom bazom podataka postoje tri slučaja anomalija:

1) write-read (WR) – T2 čita podatke objekta pre upisivanja T1
   WR konflikt, poznat i kao prljavo čitanje, je vremensko planiranje u kome transakcija T2 može da pročita objekat A baze podataka koji nije bio promenjen od transakcije T1, koja još nije potvrđena.

2) read-write (RW) - neponovljena čitanja
   Do RW konflikta bi došlo kada transakcija T1 pročita vrednost objekta A, a transakacija T2 promeni vrednost objekta A koja nije bila pročitana od transakcije T1, dok je T1 još u razvoju tj. još uvek nije potvrđena.

3) write-write (WW)
   Treći izvor nepravilnog ponašanja je mogućnost da transakcija T2 može da prepiše preko vrednosti objekta A, koja je već bila promenjena transakcijom T1, dok je T1 još u razvoju. Ako T2 ne pročita vrednost A upisanu sa T1, nastaje potencijalni problem: Pretpostavimo da su Janko i Marko dva službenika, a njihove plate moraju da se održavaju jednakim. Transakcija T1 postavlja njihove plate na $1,000, a transakcija T2 postavlja njihove plate na $2,000. Ako ih izvršavamo po serijskom uređenju da T1 sledi posle T2, oba primaju platu $2,000. Serijsko uređenje da T210 sledi posle T1 daje svakom platu $1,000. Bilo šta od ovoga je konzistentno. Zapazimo da nijedna transakcija ne čita vrednost plate pre nego što je upiše-takvo upisivanje se naziva slepo upisivanje, iz očiglednih razloga. Sada razmotrimo sledeće preplitanje akcija T1 i T2: T1 postavlja Jankovu platu na $1,000, T2 postavlja Markovu platu na $2,000, T1 postavlja Markovu platu na $1,000, i konačno T2 postavlja Jankovu platu na $2,000. Rezultat nije identičan rezultatu bilo kog serijskog izvršavanja, i vremensko preplitanje nije zbog toga serijabilno. To krši željeni konzistentni kriterijum da dve plate moraju da budu jednake.

5. Izolacija

U robustnim, kompleksnim sistemima baza podataka, mora postojati način upravljanja potencijalnih konflikata koji se mogu dogoditi prilikom pokušaja procesiranja većeg broja transakcija nad bazom podataka u isto vreme. U Oracle DBMS-u, korisnici mogu da specificiraju nivo izolacije transakcije da bi označili nivo posvoećnosti koji sistem treba da im posveti prilikom rešavanja potencijalnih konflikata.

Što je nivo izolacije veći, to će se više sistem truditi da izbegne konflikte. S druge strane, postoji cena za izbegavanje konflikata. Što je viši nivo izolacije transakije, troškovi zaključavanja mogu da se povećaju, dok istovremenost korisnika može da se smanji. Programeri i DBA moraju uzeti u obzir ove faktore kada postavljaju nivoe izolacije Oracle transakcija.

Oracle DBMS podržava tri nivoa izolacije. Postoji i četvrti nivo izolacije, *READ UNCOMMITTED*, koji nije podržan. Ovaj nivo izolacije dozvoljava konflikt – prljavo čitanje. Oracle ne koristi prljavo čitanje niti ga dozvoljava. Nivoi izolacije koje Oracle DBMS podržava su:

1) READ COMMITTED

READ COMMITTED nivo izolacije transakcije predstavlja podrazumevani nivo izolacije u Oracle-u. Sa ovim podešavanjem, svaka SQL naredba može da vidi samo informacije koje su potvrđene pre njenog početka (ne početka transakcije). Kao što smo rekli, Oracle ne dozvoljava prljavo čitanje, ali ne sprečava druge transakcije da prepišu informacije pročitane od strane posmatrane transakcije. Ako je potrebno, ovaj nivo izolacije moguće je postaviti sledećom naredbom:

SET TRANSACTION ISOLATION LEVEL READ COMMITED;

2) SERIALIZABLE

SERIALIZABLE nivo izolacije transakcije nije podržan kod distribuiranih transakcija. Sa ovim nivoom izolacije vidljive su samo informacije koje su potvrđene na početku transakcije kao i informacije stvorene u toku same transakcije korišćenjem INSERT, UPDATE i DELETE naredbi. Ovaj nivo izolacije transakcije moguće je postaviti sledećom naredbom:

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

3) READ ONLY

READ ONLY nivo izolacije transakcije je takav da omogućava vidljivost samo informacija koje su bile potvrđene pre početka transakcije. Nikakva modifikacija informacija u toku transakcije nije dozvoljena. Ovaj nivo izolacije transakcije moguće je postaviti sledećom naredbom:

SET TRANSACTION ISOLATION LEVEL READONLY;

6. Zaključavanje

Zaklučavanje je mehanizam koji sprečava destruktivne interakcije između dve transakcije koje pristupaju istom resursu – bilo to korisnički resursi kao što su tabele i redovi ili sistemski objekti nevidljvi korisniku, kao što su deljene structure podataka u memoriji. U svakom slučaju, Oracle automatski zaključava potrebne resurse kada izvršava SQL naredbe, tako da korsinik ne mora da brine o tim detaljima. Oracle takođe dozvoljava korisnicima da vrše zaklučavanja manuelno.

Da bi efektivno mogli da iskoristimo punu moć paralelnog procesiranja od ključnog značaja je razumevanje mehanizama zaključavanja. Kao korisnici Oracle baze podataka, možemo da utičemo na svaku vrstu zaključvanja kroz način na koji postavimo parametre inicijalizacije. Ako zaključavanja nisu iskorišćena efikasno, naš sistem može potrošiti toliko vremena na sinhronizaciju deljenih resursa da se ne postiže ubrzanje i povećanje veličine, čak bi sistem mogao da pretrpi degradaciju performansi.

Zaključavanje se koristi u dve glavne svrhe:

- za postizanje izolaicije transakcija i
- za postizanje keš koherencije.

Transakciono zaključavanje koristi se da implementira zaključavanje na nivou reda da bi se osigurala transakciona konzistentnost. Zaključavanje na nivou reda podržano je kako na Oracle sa jednom instancom tako i na Oracle Parallel Server-u.

Zaključavanje instanci, poznato kao i distribuirano zaključavanje, garantuje keš koherentnost. Ova vrsta zaključavanja osigurava da informacije distribuirane između više instanci koje pripadaju istoj bazi podataka ostanu konzistentne.

Zaključavanje garantuje integritet podataka dok u isto vreme dozvoljava maksimalni konkurentni pristup podacima neograničenom broju korisnika.

6.1 Automatsko zaključavanje

Automatsko zaključavanje u Oraclu se odvija od strane sistema i ne zahteva nikakvu interakciju od strane korisnika. Implicitno zaključavanje događa se kada je to potrebno za SQL naredbe, u zavistosti od zahtevane akcije.

Oracle-ov sofisticirani mehanizam zaključavanja automatski zaključava podatke u redu tabele. Oracle-ov menadžer zaključavanja održava nekoliko tipova zaključavanja reda, u zavisnosti od toga koja operacija je izvršila zaključavanje. U opštem slučaju, postoje dva tipa zaključavanja:

a) Ekskluzivno zaključavanje
   Samo jedno ekskluzivno zaključavanje se može ostvariti nad nekim resursom u isto vreme.
b) Deljeno zaključavanje
   Moguće je zaključavanje istog resursa od strane više procesa u isto vreme, ali nije moguće ekskluzivno zaklučavanje.

Oba načina zaključavanja dozvoljavaju READ operacije nad zaključanim resursom, ali brane druge operacije nad njim (kao što su UPDATE i DELETE).

6.2 Manuelno zaključavanje

U Oracle sistemu, moguće je manuelno prepisivanje (override) podrazumevane mehanizme zaključavanja. Oracle baza podataka izvršava automatsko zaključavanje da bi postigla konkurentnost i integritet podataka. Međutim, prepisivanje podrazumevanih mehanizama zaklučavanja je korisno u slučajevima kao što su:

a) Aplikacije zahtevaju konzistentnost na nivou transakcije

   U ovom slučaju, upiti moraju da proizvedu konzistentne podatke tokom trajanja transakcjije, ne odražavajući promene u drugim transakcijama. Možemo postići konzistentnost čitanja na nivou transakcije korišćenjem eksplicitnog zaključavanja, *readonly* transakcija, transakcija koje se mogu serijalizovati ili zamenom podrazumevanog zaključavanja.

b) Aplikacije zahtevaju da transakcija ima ekskluzivan pristup resursu tako da ne mora da čeka da druge transakcije završe svoje izvršenje.

Zamena Oracle-ovog automatskog zaključavanja moguća je korišćenjem neke od sledećih naredbi:

- SET TRANSACTION ISOLATION LEVEL xxxxxx;
- LOCK TABLE xxxxxx;
- SELECT xxxxx FOR UPDATE;

Zaključavanje ostvareno korišćenjem bilo koje od navedenih naredbi oslobađa se nakon završetka transakcije ili kada ih rollback do nekog savepoint-a oslobodi.

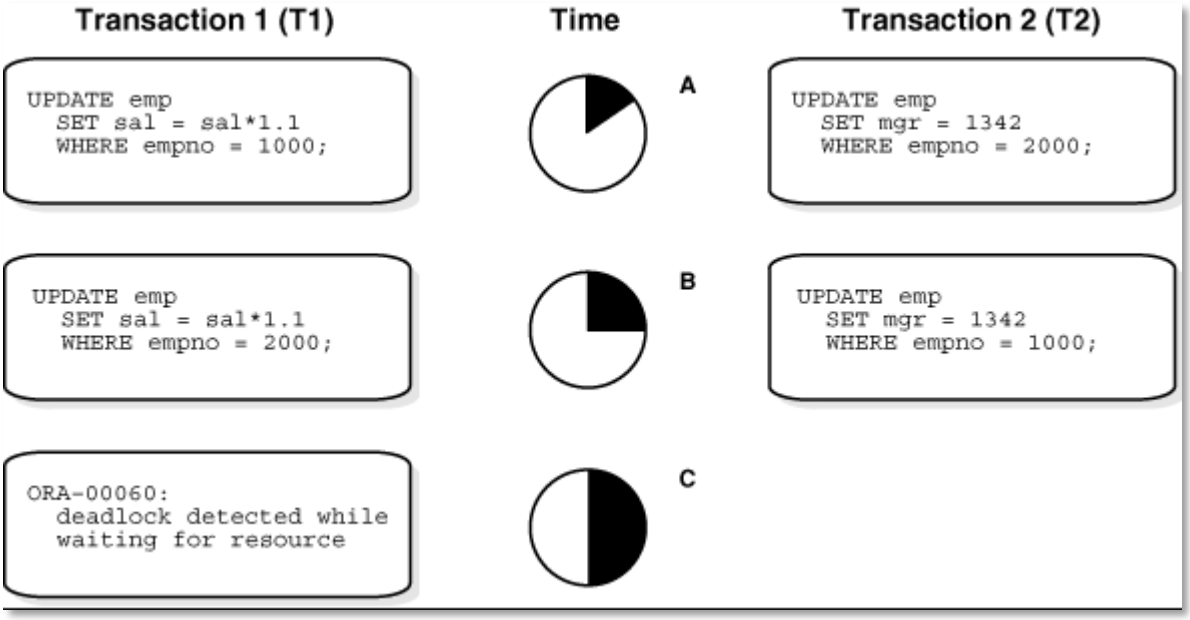
6.3 Trajanje zaključavanja

Sva zaključavanja stečena naredbama unutar transakcije čuvaju se tokom trajanja transakcije, sprečavajući destruktivne smetnje, uključujući prljava čitanja, izgubljena ažuriranja i destruktivne DDL operacije iz istovremenih transakcija. Promene napravljene od SQL naredbi jedne transakcije postaju vidljive samo drugim transakcijama koje počinju nakon što je prva transakcija potvrđena.

Oracle oslobađa sva zaključavanja stečena naredbama unutar transakcije kada se transakcija potvrdi ili poništi. Oracle takođe oslobađa zaključavanja stečena nakon tačke čuvanja kada se transakcija vrati na tačku čuvanja. Međutim, samo transakcije koje ne čekaju na prethodno zaključane resurse mogu da zaključaju sada dostupne resurse. Transakcije na čekanju će nastaviti da čekaju dok se originalna transakcija ne izvrši ili potpuno poništi.

6.4 Smrtni zagrljaj (Deadlock)

Smrtni zagrljaj nastaje kada dve ili više transakcije čekaju podatke koje su zaključali jedan od drugog. Smrtni zagrljaj sprečava transakcije da nastave da rade. Sledeća slika ilustruje dve hipotetičke transakcije u smrtnom zagrljaju:



U vremenskoj tački A ne postoji problem, pošto svaka transakcija ima zaključavanje na nivou reda u redo koji pokušava da ažurira. Svaka transakcija se nastavlja bez prekida. Međutim, u vremenskoj tački B, obe transakcije pokušavaju da ažuriraju red koji trenutno drži druga transakcija. Zbog toga dolazi do smrtnog zagrljaja u vremenskoj tački B, jer nijedna transakcija ne može da dobije resurs koji joj je potreban da nastavi ili prekine. Bez obzira koliko dugo čekale, konfliktna zaključavanja će opstati.

Oracle automatski detektuje smrtne zagrljaje i rešava ih tako što izvrši rollback jedne od transakcija koje u njemu učestvuju, otpuštajući na taj način zaključavanje koje je ta transakcija napravila i omogućavajući ostalim transakcijama da nastave sa radom. Odgovarajuća poruka prosleđena je transakciji kojoj je sistem izvršio rollback.

Do smrtnog zagrljaja najčešće dolazi kada transakcije eksplicitno prepisuju podrazumevani mehanizam zaključavanja. Međutim, s obzirom na to da Oracle ne koristi zaključavanje kada čita podatke kao i da koristi zaklučavanje na nivou reda (a ne na nivou stranice u memoriji), smrtni zagrljaju se retko dešavaju u Oracle-u.

6.5 DML zaključavanja

DML zaključavanja osiguravaju da se podaci ne mogu modifikovati u isto vreme strane više od jedne transakcija. Postoje dva tipa DML zaključavanja:

- **ROW LOCKS**

  Zaključavanje reda se ostvaruje kada se izvrši neka od INSERT, DELETE ili UPDATE naredbi ili SELECT .... FOR UPDATE naredba. Namena mu je da druge transakcija sa navedenim naredbama spreči u izvršenju. Ako UPDATE naredba jedne transakcije modifikuje neki podatak, dok u isto vreme UPDATE druge transakcije pokušava da modifikuje isti podatak, onda će druga transakcija morati da čeka da prva transakcija završi i da otpusti zaključavanje tog reda.

  SELECT naredba ne zahteva zaključavanje pa stoga može da se izvrši čak i kada neka druga transakcija zaključa red (pod uslovom da je to dozvoljeno nivoom izolacije transakcije).

  Kada se pribavi zaključavanje reda automatski se pribavlja i zaključavanje tabele. Ako je zaključavanje pribavljeno naredbom UPDATE, INSERT ili DELETE onda je zaključavanje ekskluzivno, a ako je je zaključavanje pribavljeno naredbom SELECET .... FOR UPDATE onda je deljeno.

- **TABLE LOCKS**

  Zaključavanje tabele se primarno korsiti da se osigura da tokom ažuriranja podataka ne dođe do izvrešenja DROP ili ALTER naredbe nad tabelom.

  Zaključavanje tabele ne utiče na konkurentnost DML naredbi. Konkurentnost DML naredbi određena je samo zaključavanjem reda.

6.6 Primeri zaključavanja

Zaključavanje tabele:

LOCK TABLE tables IN lock_mode MODE [WAIT [, integer] | NOWAIT];

tables – table, razdvojene zarezom
wait – Definiše koliko sekundi je dozvoljeno naredbi da čeka da bi dobila zaključavanje
nowait – Definiše da ne treba da se čeka da bi se dobilo zaključavanje

lock_mode	Značenje
ROW SHARE	Dozvoljava konkurentan pristup tabeli, ali ne dozvoljava zaključavanja cele tabele ekskluzivno
ROW EXCLUSIVE	Dozvoljava konkurentan pristup tabeli, ali ne dozvoljava zaključavanja cele tabele ekskluzivno ni zaključavanje tabele u deljivom zaključavanju
SHARE UPDATE	Dozvoljava konkurentan pristup tabeli, ali ne dozvoljava zaključavanja cele tabele ekskluzivno
SHARE	Dozvoljava konkurentan pristup tabeli, ali ne dozvoljava ažuriranje tabele
SHARE ROW EXCLUSIVE	Dozvoljava čitanje iz tabele ali brani ažuriranje tabele i zaključavanje tabele deljivim zaključavanjem
EXCLUSIVE	Dozvoljava čitanje iz tabele i brani sve ostale akcije nad tabelom

LOCK TABLE suppliers IN SHARE MODE NOWAIT;

Ova naredba bi zaključala tabelu *SUPPLIERS* deljivim zaključavanjem i ne bi čekala da se oslobodi zaključavanje koje eventualno postoji nad tom tabelom.

LOCK TABLE employee IN EXCLUSIVE MODE WAIT 5;

Ova naredba bi zaključala tabelu *EMPLOYEE* ekskluzivnim zaključavanjem i čekala bi da se oslobodi zaključavanje koje eventualno postoji nad tom tabelom ne više od 5 sekundi.

Zaključavanje reda:

select_statement FOR UPDATE [OF column_list] [NOWAIT];

select_statement – SELECT naredba koja će definisati red koji će biti zaključan
column_list – Kolone koje želimo da ažuriramo
nowait – Definiše da naredba neće čekati da bi dobila zaključavanje

```
SELECT course_number, instructor
FROM courses_tbl
FOR UPDATE OF instructor;
```

Ova naredba će nam omogućiti zaključavanje redova u tabeli *COURSES_TBL* radi ažuriranja kolone instructor. Dok ne dobijemo zaključavanje transakcija neće nastaviti sa daljim radom.

```
SELECT name, manufacturer, preference_level, sell_at_yardsale_flag
FROM my_sons_collection
WHERE hours_used = 0
FOR UPDATE;
```

Ova naredba će nam omogućiti zaključavanje redova u tabeli MY_SONS_COLLECTION gde je HOURS_USED kolona jednaka nuli. Dok ne dobijemo zaključavanje transakcija neće nastaviti sa daljim radom.

```
SELECT task, expected_hours, tools_required, do_it_yourself_flag
FROM winterize
WHERE year = TO_CHAR (SYSDATE, 'YYYY')
FOR UPDATE OF task
NOWAIT;
```

Ova naredba će nam omogućiti zaključavanje redova u tabeli WINTERIZE gde je year kolona jednaka trenutnom sistemskom vremenu. Ako ne uspemo da dobijemo zaključavanje, transakcija će nastaviti sa radom i neće čekati.

7. Literatura

[1] https://docs.oracle.com/cd/B19306_01/server.102/b14220/transact.htm

[2] https://docs.oracle.com/cd/E17984_01/doc.898/e14706/transaction_processing.htm

[3] https://docs.oracle.com/cd/E11882_01/server.112/e40540/transact.htm#CNCPT038

[4] https://docs.oracle.com/cd/E11882_01/server.112/e40540/transact.htm#CNCPT417