Univerzitet u Nišu

Elektronski fakultet

Katedra za računarstvo

Seminarski rad

**Oporavak MS SQL Server baze podataka**

****

**Student: Profesor:**

Aleksandar Mladenovic, 883 Doc. Dr Aleksandar Stanimirović

Maj 2021.

Sadržaj

[1. Uvod 3](#_Toc72699344)

[2. Pregled oporavka i kategorizacija algoritama 4](#_Toc72699345)

[2.1. Keširanje (baferovanje) blokova diskova 5](#_Toc72699346)

[2.2. Write-Ahead Logging, Steal/No-Steal, Force/No-Force 6](#_Toc72699347)

[2.3. Checkpoint 7](#_Toc72699348)

[2.4. Transaction Rollback 8](#_Toc72699349)

[3. Oporavak kod MS SQL Server 8](#_Toc72699350)

[3.1. ARIES 8](#_Toc72699351)

[3.2. Primer 11](#_Toc72699352)

[4. Zaključak 13](#_Toc72699353)

[5. Literatura 14](#_Toc72699354)

# Uvod

Sistemi baza podataka kao i bilo koji drugi računarski sistemi, podležu kvarovima, ali podaci uskladišteni u njima moraju da budu dostupni kada su potrebni i kada se zahtevaju. Kada baza podatka otkaže, ona mora da ima mogućnost za brz oporavak. Takođe mora da ima atomnost, tj ili se transakcije uspešno završe i izvrše (efekat se trajno beleži u bazi podataka) ili transakcija ne bi trebalo da ima efekat na bazu podataka. Postoje automatski i neautomatski načini za oba, pravljenje rezervnih kopija podataka i oporavak od bilo kakvih situacija otkaza. Tehnike korišćene za oporavak izgubhljenih podataka usled pada sistema, grešaka u transakcijama, virusa, katastrofalnog kvara, netačnog izvršenja naredbi i drugo, nazivaju se tehnikama oporavka baze podataka. U ovom radu biće malo više reči o tim tehnikama, kao i o tehnici koja se primenjuje na MS SQL Server bazu podataka.

# Pregled oporavka i kategorizacija algoritama

Oporavak od neuspelih transakcija obično znači da je baza podataka vraćena u najnovije konzistentno stanje neposredno pre otkaza. Da bi to uradio, sistem mora da čuva informacije o promenama koje su nastale primenom raznih transakcija na stavke podataka. Te informacije se čuvaju u datoteci evidencije*.* Kategorije oporavka mogu sažeto da se opišu na sledeće načine:

1. Ako prilikom otkaza nastane veliko oštećenje velikog dela baze podataka, kao što je pad diska, metod oporavka vraća kopiju baze podataka koja je bila smeštena u arhivsku memoriju (traka ili neki drugi vanmrežni medijum za skladištenje velikog kapaciteta) i rekonstruiše aktuelno stanje primenom operacija izvršenih transakcija iz rezervne kopije evidencije, koje su izvršene do trenutka otkaza
2. Kada baza podataka na disku nije fizički oštećena, strategija oporavka je da se identifikuje sve promene koje mogu da prouzrokuju nekonzistentnost u bazi podataka. Za ovakve kvarove nije potrebno kompletna arhivska kopija, već se podaci iz datoteke evidencije analiziraju kako bi se utvrdile odgovarajuće akcije za oporavak

Kod drugog tipa kategorija, blažih oštećenja, možemo razlikovati dve glavne tehnike: *odloženo ažuriranje (****deferred update****) i neposredno ažuriranje (****immediate update****)*. Odloženo ažuriranje ne ažurira fizički bazu podataka sve dok transakcija ne dostigne tačku izvršenja, tada se promene beleže u bazi. Pre dostizanja tačke izvršenja, sve transakcije ažuriranja su snimljene u lokalnom prostoru transakcija ili u baferima glavne memorije koju baza održava. Pre izvršenja, ažuriranja se beleže u datoteku evidencije, a zatim nakon izvršenja, ažuriranja se zapisuju u bazi na disku. Ako transakcija otkaže pre nego što dostigne tačku izvršenja, ona na bilo koji način neće da promeni bazu podataka, pa operacija *UNDO* nije potrebna. Možda će operacija *REDO* biti potrebna, da se uklone efekati operacija izvršene transakcije iz datoteke evidencije, jer se njihov efekat još nije upisao u bazu. Zato je ova tehnika poznata i kao algoritam *NO-UNDO/REDO.*

Kod tehnike neposrednog ažuriranja, baza može biti ažurirana nekim operacijama transakcije pre nego što ona dostigne svoju tačku izvršenja. Međutim, ove operacije takođe moraju da se zapisuju u datoteku evidencije prinudnim pisanjem pre nego što se primene na bazu, da bi oporavak još uvek bio moguć. Ako transakcija otkaže nakon snimanja nekoliko promena u bazi pre nego što dostigne tačku izvršenja, efekat njegovih operacija na bazu mora da se poništi, što znači da se transakcija mora vratiti nazad. U opštem slučaju, obe operacije, i *UNDO* i *REDO,* možda će biti potrebne tokom oporavka. Ova tehnika je poznata kao algoritam *UNDO/REDO*, zahteva obe operacije tokom oporavka, i najčešće se upotrebljava u praksi. U slučaju kada je potrebno zabeležiti sva ažuriranja u bazu pre nego što se transakcija izvrši, zahteva samo operaciju *UNDO*, pa se ovakav algoritam naziva *UNDO/NO-REDO* algoritam.

## Keširanje (baferovanje) blokova diskova

Proces oporavka je često usko povezan sa funkcijama operativnog sistema, posebno sa baferovanjem stranica diska baze u keš glavne memorije DBMS-a. Najčešće, veći broj stranica diska, koje uključuju stavke podataka koje treba ažurirati, se keširaju u bafer glavne memorije a zatim se ažuriraju u memoriju pre nego što opet budu napisani na disku. Keširanje stranice diskova je obično funkcija operartivnog sistema, ali zbog njegovog značaja za efikasnost postupka oporavka, DBMS upravlja njime pozivajući rutine operativnog sistema niskog nivoa.

Pogodno je razmotriti oporavak u odnosu na stranice diskova baze. Obično se kolekcija bafera u memoriji, zvana DBMS keš, drži pod kontrolom DBMS u svrhu zadržavanja tih bafera. Direktorijum za keš se koristi za praćenje koje se od stavki baze podataka nalaze u baferima. Za to se može koristiti tabela unosa. Kada DBMS zahteva akciju na nekoj stavci, prvo proverava direktorijum keš memorije da bi utvrdio da li se stranica diska koja sadrži tu stavku nalazi u keš memoriji DBMS-a. Ako nije, stavka se mora nalaziti na disku, a odgovarajuće stranice diska kopiraju se u keš memoriju. Možda će biti potrebno zameniti neke od bafera keš memorije kako bi se napravio slobodan prostor za novu savku. Neke strategije zamene su slične onima koje se koriste u operativnim sistemima, kao što su LRU (*least recently used*) ili FIFO (*first-in-first-out*), ili nove strategije specifične za DBMS kao što su DBMIN ili *Least-Likely-to-Use*.

Unosi u direktorijumu DBMS keš memorije sadrže dodatne informacije vezane za upravljanje baferom. Za svaki bafer u keš memoriji povezan je jedan prljavi (dirty) bit , koji se može uključiti u unos direktorijuma, da bi se naznačilo da li je bafer izmenjen. Kada se stranica prvo pročita sa diska baze podataka u bafer keš memorije, novi unos se ubacuje u direktorijum keš memorije sa novom adresom stranice diska, a prljhavi bit je popstavljen na nula. Čim se bafer promeni, prljavi bit za odgovarajući unos se postavlja na jedan. Dodatne informacije kao što su jedinstvene identifikacije (ID) transakcija koje modifikuju bafer se takođe mogu čuvati u direktorijumu. Kada se sadržaj bafera zamenjuje iz keš memorije, sadžaj prvo mora biti zapisan nazad na odgovarajuću stranicu diska samo ako je njegov prljavi bit jedan. Takođe je potreban još jedan bit, pin bit, koji se postavlja na jedan ako stranica u keš memoriji još uvek ne može da se zapiše nazad na disku.

Dve glavne stratategije mogu da se primene kada se modifikovani bafer vraća na disk. Prva strategija, poznata kao *ažuriranje na mestu* (***in-place updating***) zapisuje bafer na isto mesto izvornog diska, prepisujući tako staru vrednost svih promenjenih stavki podataka na disku. Dakle, održava se jedna kopija svakog bloka diska baze podataka. Ova strategija se koristi u većini sistema u praksi. Druga strategija, poznata kao *senčenje* (***shadowing***), zapisuje modifikovani bafer na različitoj lokaciji diska, tako da se može održavati više verzija stavki podataka, ali ovaj pristup se obično ne koristi u praksi. Generalno, stara vrednost stavke podataka pre ažuriranja naziva se *slika pre (****the before image - BFIM****)* a nova vrednost nakon ažuriranja naziva se *slika posle (****the after image - AFIM****).* Ako se koristi senčenje, i BFIM i AFIM mogu se čuvati na disku, stoga nije striktno neophodno voditi evidenciju oporavka.

## Write-Ahead Logging, Steal/No-Steal, Force/No-Force

Kada se koristi *ažuriranje na mestu*, neophodno je koristiti datoteku evidencije za oporavak. U ovom slučaju, mehanizam oporavka mora osigurati da je BFIM stavke podataka zabeležen u odgovarajućem zapisu datoteke i da se zapis datoteke isprazni na disk pre nego što se BFIM prepiše sa AFIM-omu bazi podataka na disku. Ovaj proces je opšte poznat kao *zapisivanje pre evidenrtiranja* (***write-ahead logging***)i neophodan je kako bi se mogla da se izvrši operacija *UNDO* ako je potrebna tokom oporavka. Pre nego što opišemo ovaj protokol, potrebno je da razlikujemo dve vrste informacija unosa datoteke uključene u naredbu za pisanje: informacije potrebne za operaciju *UNDO* i informacije potrebne za operaciju *REDO.* Prvi tip informacija uključuju staru vrednost stavke (BFIM) jer je potrebno da se poništi efekat operacije iz datoteke, postavljanjem vrednosti stavke u bazi na njen BFIM. Drugi tip informacija uključuje novu vrednost stavke (AFIM) koju je operacija napisala jer je to potrebno za ponavljanje efekta operacije iz datoteke, postavljanjem vrednosti stavke u bazi na njen AFIM. U algoritmu *UNDO/REDO* kombinuju se obe vrste zapisa. Pored toga, kada je moguće kaskadno vraćanje, pročitane stavke unosa u datoteku evidencije smatraju se da su informacije tipa *UNDO*.

Kao što je napomenuito, DBMS keš memorija drži keširane blokove diska baze podataka u glavnim memorijskim baferima, koji uključuju ne samo blokove podataka, već i blokove indeksa i blokove evidencija sa diska. Kada se evidencija napiše, ona se čuva u trenutnom baferu evidencije u keš memoriji DBMS-a. Evidencija je je jednostavno sekvencijalna disk datoteka, a DBMS keš memorija može sadržati nekoliko blokova evidencije u glavnom memorijskom baferu. Kada se završi ažuriranje blokova podataka koji su smešteni u keš DBMS-a, jedna evidencija je pridružena poslednjem baferu evidencije u keš DBMS-a. Kod pristupa zapisivanja pre evidentiranja, baferi evidencije koji sadrže povezane zapise evidencije za određeno ažuriranje blokova podataka mora prvo biti zapisan na disk pre nego što sam blok podataka može biti zapisan nazad na disk iz svog glavnog memorijskog bafera.

Standardna terminologija oporavka DBMS-a uključuje izraze *kradi/ne kradi* (***steal/no-steal***) i *prisilno/neprisilno (****force/no-force****),* koji određuju pravila koja regulišu kada se stranica iz baze podataka može zapisati na disk iz keš memorije:

1. Ako se stranica bafera keša, koju ažurira transakcija, ne može zapisati na disk pre nego što se transakcija izvrši, ovakav metod oporavka se naziva *pristup koji ne krade* (***no-steal approach***). Pinovan bit će se koristiti da naznači da li se stranica ne može zapisati nazad na disk. S druge strane, ako protokol oporavka dozvoljava pisanje ažuriranog bafera pre nego što se transakcija izvrši, to se naziva *krađom (****steal****)*. Ovakav metod se koristi kada je menadzeru DBMS keš memorije potreban okvir bafera za drugu transakciju, kao i kada zamenjuje postojeću stranicu koja je ažurirana ali čija transakcija nije izvršena. Pravilo *ne kradi* znači da operacija *UNDO* nikada neće biti potrebna tokom oporavka, jer izvršena transakcija neće imati ni jedno ažuriranje na disku pre nego što se izvrši.
2. Ako se sve stranice ažurarane transakcijom odmah upišu na disk pre nego što se transakcija izvrši, to se naziva *prisilni pristup (****force approach****)*, u suprotnom naziva se *neprisilno* (***no-force***). Pravilo *prisilno* znači da operacija *REDO* nikad neće biti potrebna tokom oporavka, jer će svaka izvršena transakcija imati sva ažuriranja na disku pre nego što se izvrši.

Tipični sistemi baza podataka koriste strategiju *kradi/neprisilno (****steal/no-force****).* Prednost pravila *kradi* je što se izbegava potreba za veoma velikim baferom za čuvanje svih ažuriranih stranica u memoriji. Prednost pravila *neprisilno* je u tome što ažurirana stranica izvršene transakcije i dalje može biti u baferu kada druga transakcija treba da je ažurira, čime se eleminišu ulazno-izlazni troškovi pisanja te stranice više puta na disk i mogućnost da mora ponovo da se pročita sa diska. To može pružiti značajnu uštedu u broju ulazno-izlaznih operacija diska kada se određena stranica intenzivno ažurira većim brojem transakcija.

## Checkpoint

Još jedan tip unosa u evidenciju je kontrolna tačka (***checkpoint***). Zapis, kog čine kontrolna tačka i lista aktivnih transakcija, se u evidenciju periodično zapisuje u trenutku kada sistem upisuje sve modifikovane DBMS bafere u bazu podataka na disku. Kao posledica toga, sve transakcije koje imaju svoje unose u evidenciju pre unosa kontrolne tačke, ne moraju da odrade ponovo operaciju *WRITE* u slučaju pada sistema, jer će sva njegova ažuriranja biti evidentirana u bazu podataka na disku tokom kontrolne tačke. Kao deo kontrolne tačke, spisak ID transakcija aktivnih transakcija u trenutku kontrolne tačke uključen je u zapis kontrolne tačke, tako da se te transakcije mogu lako identifikovati tokom oporavka.

Menadzer oporavka DBMS-a mora odlučiti u kojim intervalima uzmeti kontrolnu tačku. Interval se može meriti vremenom (npr neka merna jedinica M) ili brojem izvršenih transakcija (T) od poslednje kontrolne tačke, gde su M i T sistemski parametri. Uzimanje kontrolne tačke sastoji se od sledećih radnji:

1. Privremeno obustavljanje izvršenja transakcija
2. Prisilno se zapisuju sve glavni memorijski baferi koji su modifikovani na disku
3. Zapisivanje zapisa kontrolne tačke u evidenciju i prisilno uzapisati evidenciju na disku
4. Nastavljanje izvršenja transakcija

Kao posledica koraka 2, zapis kontrolne tačke u evidenciji može da sadrži i dadatne informacije, kao što su lista ID aktivnih transakcija i adrese prvog i najnovijeg zapisa u evidenciji za svaku aktivnu transakciju. Ovo može olakšati poništenje operacija transakcija u slučaju da se transakcija mora vratiti.

## Transaction Rollback

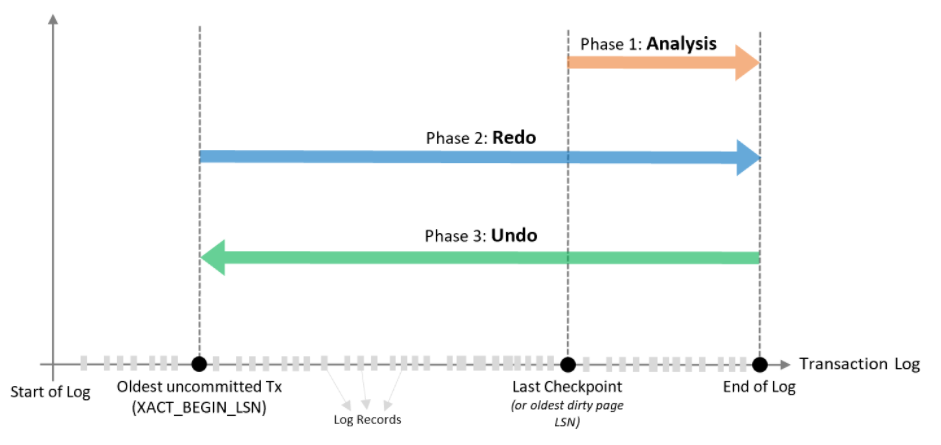
Ako transakcija ne uspe da se izvrši iz bilo kog razloga nakon ažuriranja baze podataka, biće potrebno vratiti (***roll back***) transakciju. Ako su bilo koje vrednostio stavke podataka promenjene transakcijom i zapisane u bazu, moraju se vratiti na njihove prethodne vrednosti (BFIM). Unos evidencije tipa poništen se koristi za povratak starih vrednosti stavki podataka koje se moraju vratiti.

Ako se transakcija T vrati nazad, svaka transakcija S koja je u međuvremenu pročitrala vrednost neke stavke podataka X koju je napisao T, takođe mora da se varti. Slično tome, kada se S vrati, svaka transakcija R koja je u međuvremenu pročitrala vrednost neke stavke podataka Y koju je napisao S, takođe mora da se varti, i tako nadalje. Ova pojava naziva se kaskadno vraćanje i može se desiti kada protokol oporavka osigurava rasporede oporavkaali ne osigurava stroge ili kaskadne rasporede. Kaskadno vraćanje može biti prilično složeno i dugotrajno. Zbog toga su skoro svi mehanizmi oporavka dizajnirani tako da kaskadno vraćanje nikada nije potrebno.

# Oporavak kod MS SQL Server

## ARIES

ARIES (***Algorithms for Recovery and Isolation Exploiting Semantics***) algoritam se koristi u mnogim relacionim bazama podataka, među kojima je i MS SQL Server. Ovaj algoritam koristi pristup *kradi/neprisilno (****steal/no-force****)* za pisanje, i zasniva se na tri koncepta: zapisivanje pre evidentiranja (*write-ahead logging*), ponavljanje istorije tokom operacije *REDO* i evidentiranje promena tokom operacije *UNDO*. Prvi koncept je objašnjen u ranijem delu ovog rada. Drugi koncept, ponavljanje istorije, znači da će ARIES pratiti sve radnje sisetma baze podataka pre pada, da bi rekonstruisao stanje baze podataka kada se pad dogodio. Transakcije koje se nisu završile u trenutku pada baze, aktivne transakcije, se poništavaju. Treći koncept, evidentiranje promena, sprečiće ARIES da ponavlja završene operacije poništavanja ako se tokom oporavka dogodi kvar, što uzrokuje ponovno pokretanje procesa oporavka.



Slika 1. – *ARIES algoritam kod oporavka baze*

Postupak oporavka korišćenjem ARIES algoritma sastoji se iz tri koraka: analiza, faza *REDO* i faza *UNDO*. Prvi korak, analiza, identifikuje ažurirane stranice u odgovarajućim tabelama (tabela trtansakcija i tabela ažuriranih stranica) kao i skup transakcija aktivnih u vreme pada baze. Takođe se utvrđuje odgovarajuća tačka u evidenciji gde operacija REDO treba da započne. Faza REDO zapravo pomono primenjuje ažuriranja iz evidencije u bazu podataka. Generalno, operacija REDO primenjuje se samo na izvršene transakcije, međutim to nije slučaj i kod ovog algoritma. Određene informacije u ARIES evidenciji pokazaće početnu tačku za REDO, odakle se ona primenjiju sve dok se ne stigne do kraja evidencije. Pored toga, informacije koje ARIES čuva kao i one u stranicama sa podacima omogućiće ARIES-u da utvrdi da li je operacija koju treba ponoviti već primenjena na bazu podataka i stoga je ne treba ponovo primeniti. Dakle, tokom oporavka se primenjuju samo potrebne REDO operacije. Na kraju, tokom UNDO faze evidencija se skenira unazad i operacije transakcija koje su bile aktivne u vreme pada se poništavaju obrnutim redosledom. Podaci potrebni ARIES-u da bi izveo postupak oporavka uključuju evidenciju, tabelu transakcija (*the Transaction Table*) i tabelu ažuriranih stranica (*the Dirty Page Table*). Pored toga, koristi se i kontrolna tačka (*checkpointing*). Menadzer transakcija održava ove tabele i zapisuje ih u evidenciju tokom kontrolne tačke.

U ARIES-u, svaki zapis evidencije ima pridružen broj sekvence evidencije (*LSN*) koji se monotono povećava i ukazuje na adresu zapisa evidencije na disku. Svaki taj broj odgovara određenoj akciji neke transakcije. Takođe, svaka stranica podataka čuvaće broj najnovijeg zapisa evidencije koji odgovara promeni na toj stranici. Zapis evidencije se zapisuje za bilo koju od sledećih akcija: ažuriranje stranice (*write*), izvršavanje transakcije (*commit*), prekid transakcije (*abort*), poništavanje ažuriranja (*undo*) i završavanje transakcija (*end*).

Najčešća polja u svim zapisima evidencije sadrže prethodni broj sekvence za tu transakciju, ID transakcije i tip zapisa evidencije. Prethodni broj sekvence je veoma bitan jer povezuje zapise evidencije (obrnutim redosledom) za svaku transakciju. Za akciju ažuriranja (*write*), potrebna su dodatna polja i to ID stranice koja sadrži stavku, dužinu ažurirane stavke, njegov ofset od početka stranice, slika stavke pre i posle.

Pored evidencije, za efikasan oporavak potrebne su i dve tabele: tabela transakcija i tabela ažuriranih stranica, koje održava menadzer transakcija. Kada se dogodi pad baze, ove tabele se obnavljaju u fazi analize oporavka. Tabela transakcije sadrži unos za svaku aktivnu transakciju, sa informacijama kao što su ID transakcije, status transakcije i broj sekvence najnovijeg zapisa evidencije za transakciju. Tabela ažuriranih stranica sadrži unos za svaku ažuriranu stranicu u baferu, koji sadrži ID stranice i broj sekvence koji odgovara najranijem ažuriranju te stranice.

Kontrolna tačka u ARIES-u sastoji se od sledećeg: zapisivanje zapisa početne kontrolne tačke u evidenciju, zapisivanje zapisa poslednje kontrolne tačke u evidenciju i zapisivanje broja sekvence zapisa početne kontrolen tačke u posebnu datoteku. Ovoj posebnoj datoteci se pristupa tokom oporavka da bi se pronašle informacije o poslednjoj kontrolnoj tački. Pored zapisa poslednje kontrolne tačke, sadržaj obe tabele, i tabele transakcija i tabele ažuriranih stranica, dodaje se na kraj evidencije. Da bi se smanjio trošak, koristi se „*nejasna kontrolna tačka*“ (***fuzzy checkpointing***), tako da DBMS može da nastavi da izvršava transakcije tokom kontrolne tačke. Pored toga, sadržaj DBMS keša ne mora da se ISPIRE na disk tokom kontrolne tačke, jer tabela transakcija i tabela ažuriranih stranica koje se nalaze u evidenciji, sadrže informaicje potrebne za oporavak. U slučaju da se pad dogodi tokom kontrolne tačke, posebna datoteka upućuje na prethodnu kontrolnu tačku koja se koristi za oporavak.

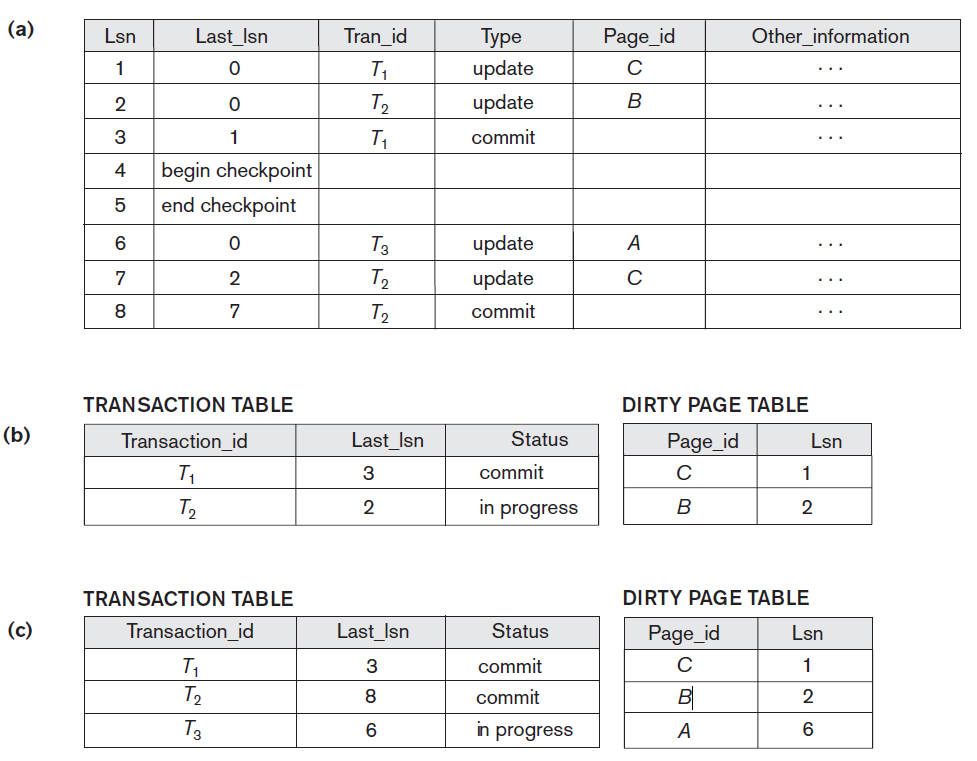
Kada se desi pad, pokreće se ARIES menadzer oporavka. Prvo se pristupa informacijama sa poslednje kontrolne tačke preko posebne datoteke. Faza analize započinje sa zapisom početne kontrolne tačke i nastavlja sve do kraja evidencije. Kada se dođe do zapisa poslednje kontrolne tačke, pristupa se tabeli transakcija i tabeli ažuriranih stranica (koje su zapisane u evidenciji tokom faze kontrolnih tačaka). Tokom analize, zapisi evidencija koji se analiziraju mogu da prouzrokuju modifikaciju ove dve tabele. Na primer, ako je nađen poslednji zapis evidencije za transakciju T1 u tabeli transakcija, tada se unos za T1 briše iz te tabele. Ako se nađe neka druga vrsta zapisa evidencije za transakciju T2, tada se unos za T2 ubacuje u tabelu transakcija, ako već nije prisutan a poslednje polje koje sadrži broj sekvence se menja. Ako zapis evidencije odgovara promeni za stranicu P, tada će se izvršiti unos za stranicu P, ako nije nađen u tabeli, a poveyano polje broja sekvence će se pormeniti. Kada je faza analize završena, potrebne informacije za fazu REDO i UNDO sastavljene su u tabelama.

Zatim sledi faza REDO. Da bi smanjio količinu nepotrebnog posla, ARIES počinje da se izvršava od tačke u evidenciji od koje zasigurno zna da su prethodne promene ažuriranih stranica već primenjene na bazu podataka na disku. To može da utvrdi pronalaženjem najmanjeg broja sekvence M, od svih ažuriranih stranica u tabeli ažuriranih stranica, što ukazuje na položaj evidencije odakle ARIES treba da započne REDO fazu. Sve promene koje odgovaraju jednačini: broj sekvence < M, za transakcije koje treba da se ponove, mora da su već prenete na disk ili su već zamenjene u baferu i tabeli ažuriranih stranica. Dakle, faza REDO započinje od zapisa evidencije čiji je broj sekvence jednak M, i skenira sve do kraja evidencije. Za svaku promenu zabeleženu u evidenciji, REDO algoritam proverava da li se promena mora ponovo primeniti. Na primer, ako se promena zabeležena u evidenciji odnosi na stranicu P koja se ne nalazi u tabeli ažuriranih stranica, onda je ta promena već na disku i ne treba je ponovo primenjivati. Ili, ako se promena zabeležena u evidenciji (npr. broj sekvence = N) odnosi na stranicu P, a tabela ažuriranih stranica sadrži unos za P sa brojem sekvence veće od N, tada je promena već prisutna. Ako ne ispunjava ni jedan od ova dva uslova, stranica P se čita sa diska a broj sekvence sačuvan na toj stranici, upoređuje se sa N. Ako je manji od N, tada je promena primenjena i stranicu nije potrebno ponovo zapisati na disk.

Kada se REDO faza završi, baza podataka je u istom onom stanju kao i kada je bila pre nego što je došlo do pada. Skup aktivnih transakcija je identifikovan u tabeli transakcija tokom faze analize. Sada se u fazi UNDO nastavlja sa skeniranjem unazad od kraja evidencije i poništava odgovarajuće akcije. Za svaku akciju koja je poništena zapisuje se kompenzacioni zapis evidencije. UNDO čita unazad evidenciju dok se ne poništi svaka akcija skupa aktivnih transakcija. Kada se ovo završi, postupak oporavka je završen i normalna obrada može početi ponovo.

## Primer

Razmotrimo jedan oporavak baze prikazan na slici 1. Tamo su tri transakcije: T1, T2 i T3, gde T1 ažurira stranicu C, T2 ažurira stranicu B i C, a T3 ažurira stranicu A.



Slika 2. – *Primer oporavka baze korišćenjem ARIES algoritma*

1. *Evidencija u trenutku pada baze*
2. *Tabela transakcija i tabela ažuriranih stranica u trenutku kontrolne tačke*
3. *Tabela transakcija i tabela ažuriranih stranica nakon faze analize*

Tabela (a) na slici prikazuje parcijalne sadržaje evidencije, a tabela (b) prikazuje sadržaje tabele transakcija i tabele ažuriranih stranica. Zamislimo da se otkaz desio u tom trenutku. Kada se dođe do kontrolne tačke, preuzima se adresa povezanog zapisa početne kontrolne tačke, a to je lokacija 4. Faza analize počinje od lokacije 4 sve dok ne stigne do kraja. Zapis poslednje kontrolne tačke (lokacija 5) sadržao bi tabelu transakcija i tabelu ažuriranih stranica koje su prikazane u tabeli (b), a faza analize će dalje rekonstruisati ove tabele. Kada faza analize naiđe na zapis evidencije 6, u tabeli transakcija vrši se novi unos za transakciju T3,a u tabeli ažuriranih stranica vrši se novi unos za stranicu A. Nakon što je analiziran zapis evidencije 8, status transakcije T2 u tabeli transakcija se menja u *izvršen (committed)*. Tabela (c) prikazuje stanje tabela nakon faze analize. Za *REDO* fazu, najmanji broj sekvence evidencije (*LSN*) u tabeli ažuriranih stranica je 1. Stog *REDO* faza započinje od zapisa evidencije 1 i nastavlja sa *REDO* ažuriranjima. Niz brojeva sekvenci {1, 2, 6, 7} koji odgovaraju ažuriranim stranica redosledom C, B, A i C, nisu niži od brojeva sekvenci tih stranica. Dakle, te stranice podataka čitaće se ponovo a ažuriranja primeniće ponovo iz evidencije (pod pretpostavkom da su aktuelni brojevi sekvenci uskladišteni na tim stranicama podataka manji od odgovarajućeg unosa evidencije). U ovom trenutku, faza *REDO* je završena i faza *UNDO* započinje. Iz tabele transakcija (tabela (c)), *UNDO* se primenjuje samo na aktivnu transakciju T3. *UNOD* faza započinje unosom evidencije 6 (poslednje ažuriranje za T3) i nastavlja se unazad u evidenciju. Povratni lanac ažuriranja za transakciju T3 (samo zapis evidencije 6 u ovom primeru) se pronalazi i poništava.

# Zaključak

Menadžer oporavka DBMS-a odgovoran je za osiguranje valentnosti i trajnosti transakcija. Osigurava valentnost poništavanjem radnji transakcija koje se ne izvršavaju i trajnost osiguravajući da se sve radnje izvršenih transakcija prežive otkaze i kvarove medija. Nakon otkaza DBMS se ponovo pokreće. Menadžeru oporavka je data kontrola i mora bazu podataka dovesti u doslednjo stanje. On je takođe odgovoran za poništavanje radnji prekinute transakcije.

ARIES je algoritam oporavka dizajniran da radi sa *steal/no-force* pristupom. Kada se menadžer oporavka pozove nakon pada, ponovno pokretanje se nastavlja u tri faze. U fazi analize identifikuju se modifikovane stranice i aktivne transakcije u trenuitku pada. U *REDO* fazi ponavlja se sve radnje, počev od odgovarajuće tačke u evidenciji, i vraća stanje baze podataka u ono što je bilo u trenutku pada. Konačno, u faza *UNDO* poništava radnje transakcije koje nisu počinjene, tako da baza podataka održava samo radnje izvršenih transakcija.

# Literatura

1. Ramez Elmasri and Shamkant Navathe. 2010. Fundamentals of Database Systems (6th. ed.). Addison-Wesley Publishing Company, USA.
2. Raghu Ramakrishnan and Johannes Gehrke. 2002. Database Management Systems (3rd. ed.). McGraw-Hill, Inc., USA.
3. C. Mohan, Don Haderle, Bruce Lindsay, Hamid Pirahesh, Peter Schwarz. ACM Transactions on Database Systems, Vol 17, No. 1, March 1992, Pages 94-162
4. <https://en.wikipedia.org/wiki/Microsoft_SQL_Server>
5. <https://en.wikipedia.org/wiki/Algorithms_for_Recovery_and_Isolation_Exploiting_Semantics>
6. <https://docs.microsoft.com/en-us/sql/relational-databases/backup-restore/restore-and-recovery-overview-sql-server?view=sql-server-ver15>
7. <https://www.w3schools.in/dbms/data-recovery-in-dbms/>