Predavanje XIII.

Sustav oporavka

BAZE PODATAKA II

doc. dr. sc. Goran Oreški
Fakultet informatike,
Sveučilište Jurja Dobrile, Pula

Sadržaj

- ponavljanje prethodnih predavanja
  - kontrola istodobnosti
  - lock-based protokol
  - dvo-fazni protokol zaključavanja
  - zastoji
  - Timestamp-Based protokoli
  - Validation-Based protokoli
  - Snapshot Isolation
- sustav oporavka
- klasifikacija kvarova
- pohrana
  - implementacija trajne pohrane
  - pristup podacima
- oporavak i atomičnost
  - log zapisi
  - checkpoints
- algoritmi oporavka
- udaljeni backup

Ponavljanje

- lock-based protokol
- zaključavanje je mehanizam za kontrolu istodobnog pristupa podatkovnim elementima
- podatkovni elementi mogu biti zaključani na dva načina:
  - exclusive (X) mode
  - shared (S) mode
- dvo-fazni protokol zaključavanja
  - faza 1: rastuća faza
  - faza 2: silazna faza

Ponavljanje

- deadlock (zastoj) - niti jedna transakcija ne može nastaviti izvođenje
- prevencija, detekcija i oporavak od zastoja
- višestruka granularnost
- timestamp protokol (*engl. timestamp-ordering protocol*) osigurava da se bilo koja konfliktna *read* i *write* operacija izvršava po timestamp redoslijedu
- validation-based protokol
  - čitanje i izvršavanje
  - validacija
  - pisanje

Ponavljanje

- Snapshot Isolation (SI)
- transakcije se izvode koristeći snapshot isolation:
  - prilikom pokretanja transakcija uzima snimku potvrđenih podataka
  - uvjek čita/modificira podatke u svojoj vlastitoj snimci
  - izmjene nisu vidljive u istodobnim transakcijama
  - zapisivanje se završava s potvrdom
- first-committer-wins pravilo:
  - potvrđuje se transakcija samo ako niti jedna istodobna transakcija nije zapisivala podatke koji se žele izmijeniti
- prednosti i nedostatci SI

Sustav oporavka

- računalni sustav, kao i bilo koji drugi uređaj, može pretrpiti kvar iz različitih razloga:
  - kvar diska,
  - nestanak struje,
  - greška software-a,
  - vatra u server sobi,
  - sabotaža…
- u slučaju bilo kakvog kvara, informacije mogu biti izgubljene!
- sastavni dio sustava baze podataka je **shema oporavka** (SO) koja može vratiti bazu podataka u konzistentno stanje koje je postojalo prije kvara

Sustav oporavka

- potrebno je napraviti određene korake unaprijed da bi se osiguralo očuvanje svojstava ***atomičnosti*** **i** ***trajnosti transakcije***
- shema oporavka mora podržavati visoku razinu dostupnosti (engl. high availability)
  - baza mora biti dostupna za korištenje **jako visoki** postotak vremena
  - što se smatra **jako visokim** postotkom?

Sustav oporavka

Availability %	Downtime per year	Downtime per month	Downtime per week	Downtime per day
90% ("one nine")	**36.53 days**	**73.05 hours**	**16.80 hours**	**2.40 hours**
95% ("one and a half nines")	18.26 days	36.53 hours	8.40 hours	1.20 hours
97%	10.96 days	21.92 hours	5.04 hours	43.20 minutes
98%	7.31 days	14.61 hours	3.36 hours	28.80 minutes
99% ("two nines")	**3.65 days**	**7.31 hours**	**1.68 hours**	**14.40 minutes**
99.5% ("two and a half nines")	1.83 days	3.65 hours	50.40 minutes	7.20 minutes
99.8%	17.53 hours	87.66 minutes	20.16 minutes	2.88 minutes
99.9% ("three nines")	**8.77 hours**	**43.83 minutes**	**10.08 minutes**	**1.44 minutes**
99.95% ("three and a half nines")	4.38 hours	21.92 minutes	5.04 minutes	43.20 seconds
99.99% ("four nines")	**52.60 minutes**	**4.38 minutes**	**1.01 minutes**	**8.64 seconds**
99.995% ("four and a half nines")	26.30 minutes	2.19 minutes	30.24 seconds	4.32 seconds
99.999% ("five nines")	**5.26 minutes**	**26.30 seconds**	**6.05 seconds**	864.00 milliseconds
99.9999% ("six nines")	**31.56 seconds**	**2.63 seconds**	**604.80 milliseconds**	**86.40 milliseconds**
99.99999% ("seven nines")	**3.16 seconds**	**262.98 milliseconds**	**60.48 milliseconds**	**8.64 milliseconds**
99.999999% ("eight nines")	**315.58 milliseconds**	**26.30 milliseconds**	**6.05 milliseconds**	864.00 microseconds
99.9999999% ("nine nines")	**31.56 milliseconds**	**2.63 milliseconds**	**604.80 microseconds**	**86.40 microseconds**

Izvor: https://en.wikipedia.org/wiki/High_availability

Sustav oporavka

- stoga je potrebno napraviti određene korake unaprijed da bi se osiguralo očuvanje svojstava atomičnosti i trajnosti transakcije
- shema oporavka mora podržavati visoku razinu dostupnosti (engl. high availability HA)
  - baza mora biti dostupna za korištenje jako visoki postotak vremena
  - što se smatra visokim postotkom?
- SO mora podržavati sposobnost održavanja sigurnosne kopije sinkronizirane sa sadržajem primarne kopije baze
  - da bi podržala HA u slučaju kvara (ili planiranog gašenja servera za nadogradnju HW\SW i održavanje)

Klasifikacija kvarova

- postoje različiti tipovi kvarova koji se mogu dogoditi, svaki od kojih traži drugačije upravljanje
- u nastavku predavanja bavit ćemo se samo slijedećim tipovima kvarova:
  - **transakcijski kvar**: dvije vrste grešaka
    - logička greška – transakcija ne može nastaviti normalno izvođenje zbog nekog unutarnjeg problema, kao što je: pogrešan unos, podaci nisu pronađeni, overflow, prekoračeni limit resursa…
    - pogreška sustava – sustav je ušao u nepoželjno stanje (npr. deadlock), što znači da transakcija ne može nastaviti normalno izvođenje
      - transakcija može biti ponovno pokrenuta naknadno

Klasifikacija kvarova

- u nastavku predavanja bavit ćemo se samo slijedećim tipovima kvarova:
  - **pad sustava** – kvar hardware-a ili bug software-a baze ili operativnog sustava, koji uzrokuje gubitak sadržaja nestalnih medija pohrane (engl. volatile storage) i zaustavlja procesiranje transakcije
    - sadržaj stalnih medija (engl. non-volatile storage) ostaje isti i nije iskvaren
    - pretpostavka da greške hardware-a i bug-ovi software-a zaustavljaju sustav ali ne kvare sadržaj stalnih medija je poznat pod nazivom ***fail-stop assumption***
    - dobro dizajnirani sustavi imaju brojne unutarnje provjere na hardware i software razini koje zaustavljaju sustav kada se dogodi greška
      - stoga je fail-stop pretpostavka razumna
  - **kvar medija za pohranu –** mehanički kvar diska ili kvar tijekom operacije prijenosa podataka

Klasifikacije kvarova

- da bi odredili kako bi se sustav trebao opraviti od kvara, potrebno je identificirati modove kvara (*engl. failure modes*) uređaja koji služe za pohranu podataka
  - potom razmotriti kako ti modovi utječu na sadržaj baze podataka…
  - i definirati algoritme koji će osigurati konzistentnost baze podataka i atomičnost transakcije
- algoritmi, poznati kao i algoritmi oporavka (engl. recovery algorithms), se sastoje od dva dijela:
  - akcije poduzete tijekom normalnog procesiranja transakcije
    - da bi osiguralo dovoljno informacija za oporavak
  - akcije nakon što se kvar dogodio
    - povrat sadržaja baze podataka u stanje koje osigurava konzistentnost, atomičnost i trajnost

Pohrana

- podatci mogu biti pohranjeni na različitim medijima
  - mediji se razlikuju po brzini, kapacitetu, otpornosti na kvarove
- možemo identificirati tri kategorije pohrane:
  - nestalna pohrana (engl. Volatile storage)
  - stalna pohrana (engl. Non-Volatile storage)
  - trajna pohrana (engl. Stable storage)
- **trajna pohrana**, ili preciznije, njena aproksimacija, igra kritičnu ulogu u algoritmima oporavka

Implementacija trajne pohrane

- RAID sustavi garantiraju da kvar jednog diska neće rezultirati gubitak podataka
  - čak i prilikom transfera podataka
  - ali ne od prirodnih katastrofa…
    - u tu svrhu se koriste udaljene kopije podataka (*engl. remote backup*) koje ćemo istražiti u nastavku
- potrebno je replicirati potrebne podatke na više medija stalne pohrane
  - u pravilu diskova s nezavisnim modovima kvara
  - ažurirati podatke na kontrolirani način da bi se izbjegla šteta na podatcima u slučaju *kvara prilikom transfera*

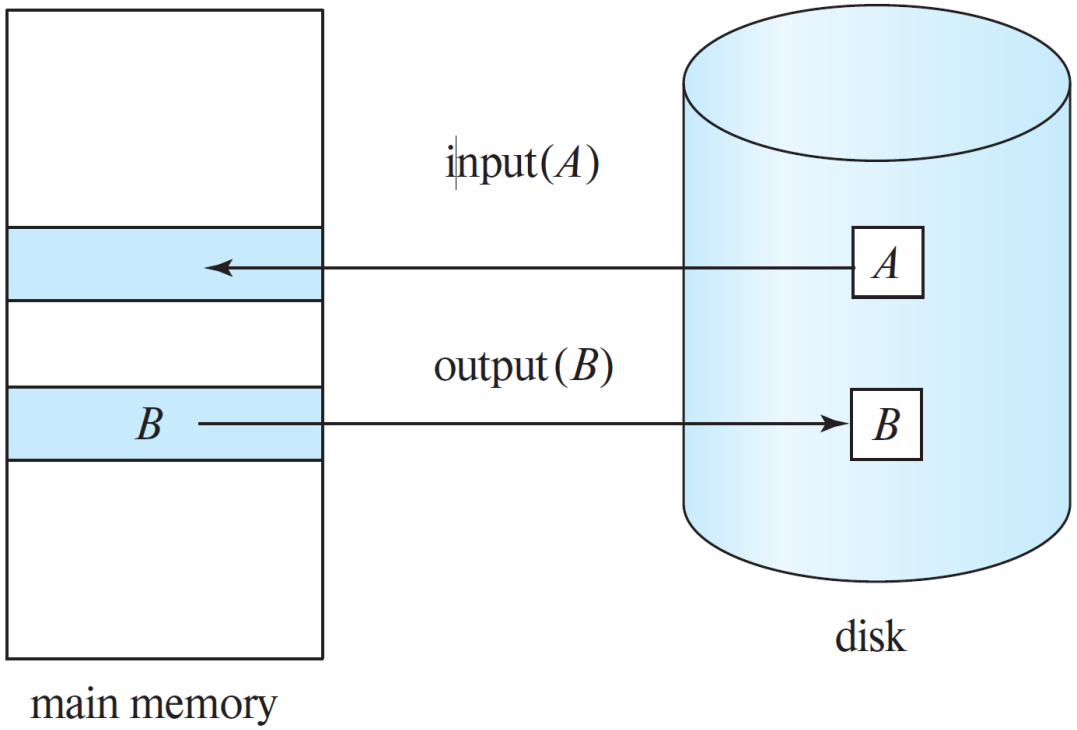
Implementacija trajne pohrane

- transfer bloka između memorije i diska može rezultirati:
  - uspješnim prijenosom
  - djelomičnim neuspjehom
    - greška tijekom prijenosa
  - potpunim neuspjehom
    - greška prije prijenosa
- u slučaju neuspjeha, sustav treba detektirati stanje i pokrenuti proceduru oporavka
  - sustav mora čuvati dva fizička bloka za svaki logički blok baze podataka
- podaci se zapisuju na prvi fizički blok, potom na drugi
  - operacija je gotova tek kada je drugo pisanje uspješno obavljeno

Pristup podacima

- I/O operacije se izvršavaju u blokovima
  - blok na disku - *fizički blok*
  - blok privremeno u glavnoj memoriji - *buffer blok*
  - područje memorije gdje su blokovi privremeno pohranjeni – *disk buffer*
- prenošenje blokova između GM i diska su inicirane kroz slijedeće dvije operacije:
  1. ***input(B)*** prebacuje fizički blok B u GM
  2. ***output(B)*** prebacuje buffer blok B na disk i mijenja odgovarajući fizički blok na disku

Pristup podacima



Pristup podacima

- svaka transakcija ima **privatan dio** memorije u kojem se drže podaci koje ista koristi
  - stvara sustav prilikom inicijalizacije transakcije
  - otpušta ga nakon commit-a ili abort-a
- interakcija s bazom podataka:
  1. *read(X)* dodjeljuje vrijednost podatka *X* lokalnoj varijabli x_i, na način:
     I. ako blok B_X na kojem se *X* nalazi nije u glavnoj memoriji, izdaje input(B_X)
     II. dodjeljuje x_i vrijednost *X* iz buffer bloka
  2. *write(X)* dodjeljuje vrijednost lokalne varijable x_i podatku *X* u buffer bloku
     I. ako blok $B_{X,}$ na kojem se *X* nalazi nije u glavnoj memoriji, izdaje input(B_X)
     II. dodjeljuje vrijednost iz x_i u *X* u buffer B_X

Oporavak i atomičnost

- sjetimo se transakcije prebacivanja novca s jednog računa banke na drugi
- najčešće korištena tehnika za oporavak se bazira na **log zapisima** (*engl. log-based recovery*)
- **log** je niz log zapisa kojima se bilježe update aktivnosti na bazi podataka
- postoji nekoliko log zapisa; ***update log zapis*** opisuje jedno pisanje (*write*) baze podataka a sadrži:
  - identifikator transakcije, identifikator podatka, staru vrijednost i novu vrijednost

Log zapisi

- log zapis ćemo obilježavati s $<T_i, X_j, V_1, V_2>$
  - transakcija T_i je izvela operaciju *write* na podatku X_j, koji je imao vrijednost V_1 prije nego je postavljena vrijednost V_2
- još neki log zapisi koje ćemo koristiti:
  - $<T_i\ start>$
  - $<T_i\ commit>$
  - $<T_i\ abort>$
- kada transakcija izvodi *write*, bitno je da se prvo zapiše log, a potom modificira baza
- log se mora voditi na trajnoj pohrani

Modifikacija baze podataka

- koraci transakcije u modificiranju podatka:
  - transakcija izvodi neku operaciju u privatnom dijelu glavne memorije
  - transakcija modificira blok u disk bufferu (u GM) koji sadrži taj podatak
  - sustav baze podataka izvršava output operaciju kojom zapisuje blok na disk
- modifikacija baze podataka se smatra ukoliko je napravljen update na disk buffer-u ili samom disku
  - update na privatnom dijelu transakcije se ne smatra modifikacijom
- transakcija može koristiti dvije tehnike:
  - *deferred-modification* – modifikacija baze podataka se radi nakon commita
  - *immediate-modification* – modifikacija se događa dok je transakcija aktivna

Modifikacija baze podataka

- algoritam oporavka mora uzeti u obzir puno faktora, uključujući:
  - mogućnost da je transakcija napravila commit iako neke modifikacije postoje samo na disk buffer-u
  - mogućnost da je transakcija modificirala bazu podataka u aktivnom stanju i da, kao rezultat kvara, mora napraviti abort
- sve modifikacije moraju biti unaprijed zapisane na log-u, stoga sustav ima staru i novu vrijednost, te može napraviti:
  - *undo* operaciju
    - postavljanje podataka na staru vrijednost iz log zapisa
  - *redo* operaciju
    - postavljanje podataka na novu vrijednost iz log zapisa

Kontrola istodobnosti i oporavak

- algoritmi oporavka u pravilu zahtijevaju:
  - ukoliko je podatak mijenjan od transakcije, niti jedna druga transakcija ne može mijenjati podatak prije commita prve
  - dvo-fazni protokol zaključavanja
    - immediate modification, deferred modification
  - ili snapshot isolation, validation-based protocol
    - deferred modification

<T_0 start>
<T_0, A, 1000, 950>
<T_0, B, 2000, 2050>
<T_0 commit>
<T_1 start>
<T_1, C, 700, 600>
<T_1 commit>

Commit transakcije

- transakcija je commit-irana kada je *commit log zapis* pohranjen na trajnoj pohrani
  - u tom trenutku svi ostali log zapisi su također pohranjeni na trajnoj pohrani
- stoga sustav ima dovoljno informacija u log-u, čak i da se dogodi kvar, da sve promjene može ponovno izvesti

- dvije transakcije, primjer s prošlih predavanja:

T_0: read(A);
$A := A - 50$;
write(A);
read(B);
$B := B + 50$;
write(B).

T_1: read(C);
$C := C - 100$;
write(C).

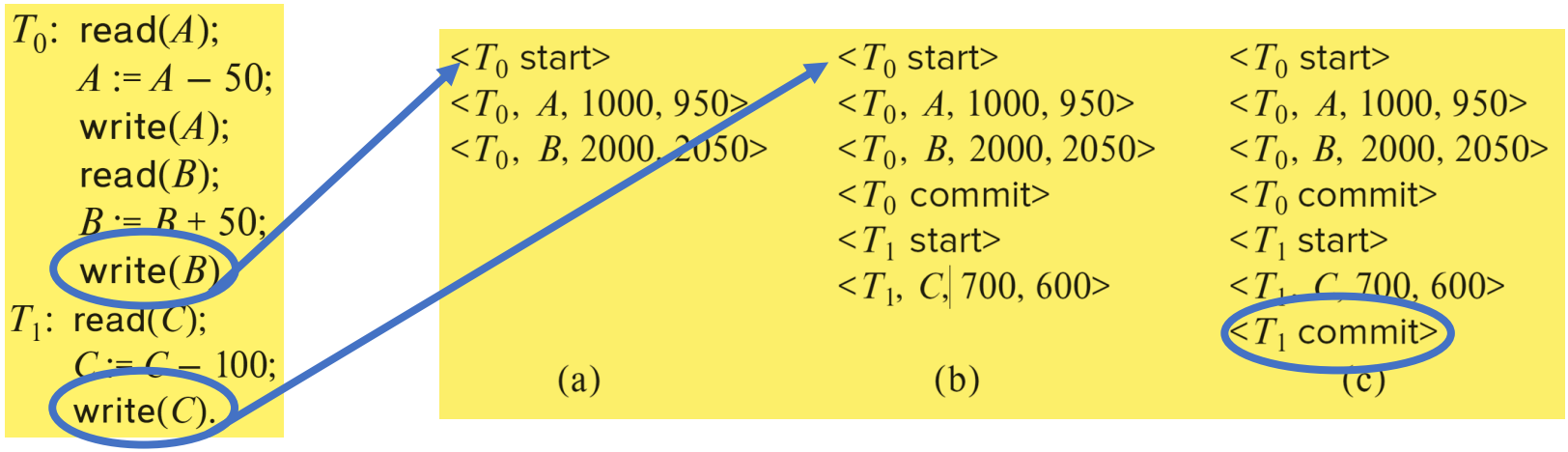
Redo i Undo transakcije

- log za prethodne transakcije
- shema oporavka koristi dvije procedure
  - obje koriste log
- *redo(T_i)*
  - postavlja sve vrijednosti podataka na *new*
  - redoslijed update je bitan
  - slijedno skeniranje log-a
- *undo(T_i)*
  - postavlja sve vrijednosti podataka na old
  - kompliciraniji postupak, redoslijed bitan
  - moraju se zapisivati ***redo-only*** log zapisi

Log	Database
$<T_0$ start$>$	
$<T_0, A, 1000, 950>$	
$<T_0, B, 2000, 2050>$	
	$A = 950$
	$B = 2050$
$<T_0$ commit$>$	
$<T_1$ start$>$	
$<T_1, C, 700, 600>$	
	$C = 600$
$<T_1$ commit$>$	

Redo i Undo transakcije

- kada se dogodi pad sustava, provjerava se log radi utvrđivanja koju operaciju treba provesti radi ostvarivanja atomičnosti
  - transakcija treba *undo* ukoliko u log-u postoji <T_i *start*> ali ne <T_i *commit*> ili <T_i *abort*>
  - transakcija treba *redo* ukoliko u log-u postoji <T_i *start*> i <T_i *commit*> ili <T_i *abort*>



Checkpoints

- dva velika problema s navedenim pristupom:
  - proces pretrage po log-u je vremenski zahtjevan
  - većina transakcija, koje prema algoritmu, trebaju napraviti *redu* su već napravile update baze podataka
- da bi smanjili nepotrebne korake, koriste se kontrolne točke (*engl.* ***checkpoints***)
- checkpoint shema:
  - ne dopušta nikakav update dok se izvodi checkpoint operacija
  - zapisuje sve modificirane buffer blokove na disk kada se izvede checkpoint

Checkpoints

- checkpoint se izvodi na način:
  - na trajnu pohranu se zapisuju svi log zapisi iz glavne memorije
  - zapisuju se svi modificirani buffer blokovi na disk
  - na trajnu pohranu se zapisuje log zapis <*checkpoint L*>, gdje je *L* lista svih aktivnih transakcija u vrijeme checkpoint-a
- nakon pada, sustav može unazad pretražiti log dok ne dođe do <*checkpoint L*> zapisa
- u skupu transakcija *T:*
  - *redo* i *undo* operacije se izvode samo na transakcijama iz *L* i onima koje su poslije checkpoint-a započele
  - sve T_k iz *T* koje nemaju <T_k *commit*> ili <T_k *abort*> u logu, izvodi se *undo*(T_k)
  - sve T_k iz *T* koje imaju <T_k *commit*> ili <T_k *abort*> u logu, izvodi se *redo*(T_k)

Algoritmi oporavka

- identificirali smo transakcije nad kojima se izvršava undo i redo operacija
  - ali nismo definirali točan algoritam tih operacija
- algoritam oporavka koristi:
  - log zapise za oporavak od transakcijskog kvara
  - kombinaciju zadnjeg checkpoint-a i log zapisa za oporavak od pada sustava
- algoritam opravka koji će biti opisan podrazumijeva da podatak koji je izmijenjen od transakcije koje nije napravila commit ili abort, ne može biti mijenjan od strane bile koje druge transakcije

Rollback transakcije

- rollback transakcije T_i tijekom normalne operacije
  - ne kao oporavak od pada sustava

1. log se prolazi od kraja, i za svaki pronađeni log zapis T_i u formi $<T_i, X_j, V_1, V_2>$
   a. vrijednost V_1 se zapisuje u X_j,
   b. poseban **redo-only** log zapis $<T_i, X_j, V_1>$ se dodaje u log, gdje je V_1 vraćena vrijednost podatka X_j tijekom rollback-a
2. kada se dohvati $<T_i\ start>$, obilazak log-a se zaustavlja i dodaje se log zapis $<T_i\ abort>$

- sve operacije transakcije ili u ime transakcije su evidentirane na log-u

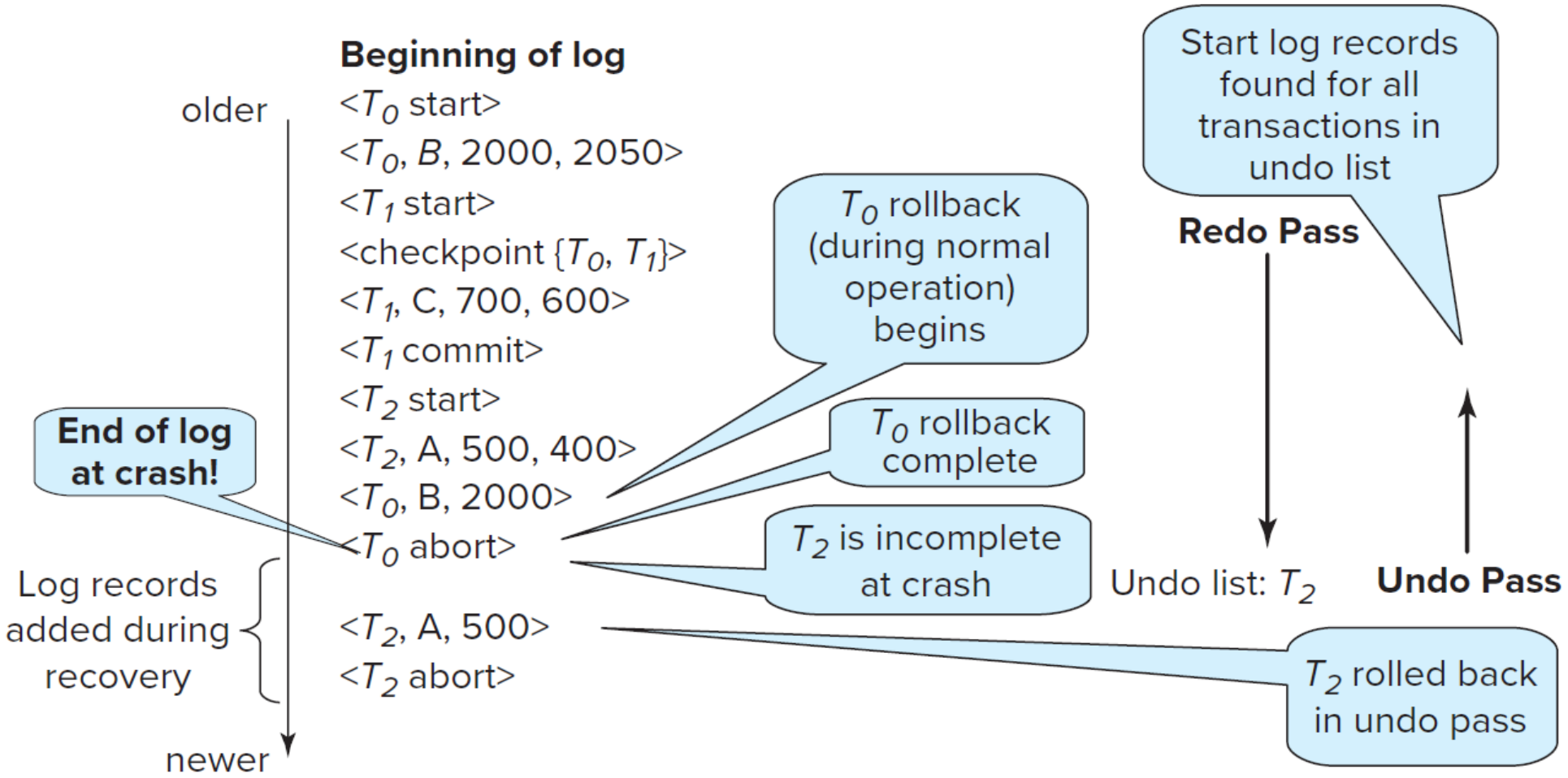
Rollback transakcije

- oporavak, nakon pada sustava, se odvija u dva koraka:
  - redo korak
    a. lista transakcija za rollback, undo-lista, se inicijalno postavlja na listu *L* iz zadnjeg <*checkpoint L*> log zapisa
    b. kada se dohvati log zapis u formi $<T_i, X_j, V_1, V_2>$ ili $<T_i, X_j, V_1>$, napravi se redo operacije, tj. V_2 se upisuje na X_j
    c. kada se pronađe zapis $<T_i\ start>$, T_i se dodaje na undo-listu
    d. kada se pronađe zapis $<T_i\ abort>$ ili $<T_i\ commit>$, T_i se briše s undo-liste
  - na kraju redo koraka, undo-lista sadrži listu svih transakcija koje su nepotpune
    - nije napravljen commit
    - niti je završen rollback prije pada sustava

Rollback transakcije

- undo korak
  - sustav koristi undo-listu za rollback transakcija, prolazi log od kraja
  a. kada se pronađe log zapis koji pripada transakciji s undo-liste, izvodi se undo operacija
  b. kada sustav pronađe <T_i *start*> za transakciju T_i u undo-listi, zapisuje <T_i *abort*> u log i briše T_i iz undo-liste
  c. undo korak prestaje kada undo-lista postane prazna, tj. sustav pronađe <T_i *start*> za sve transakcije koje su inicijalno bile undo-listi
- nakon undo koraka oporavak završava
  - normalna obrada transakcija nastavlja
- pojednostavljena **ARIES** metoda za oporavak

Rollback transakcije



Pad medija stalne pohrane

- gubitak podataka se može dogoditi i u stalnoj pohrani
  - rijetko se događa, ali događa
  - sustav mora biti otporan i na takvu vrstu kvarova
- osnovna shema je prebaciti (*eng. dump*) periodički sve podatke u trajnu pohranu
  - npr. svaki dan, na magnetnu traku
  - u slučaju pada koristi se zadnja verzija za vraćanje podataka
    - nakon što se podaci prebace koristi se log, za obnovu zadnjeg stanja
  - jedan od pristupa prebacivanja podataka zahtjeva da nema aktivne transakcije za vrijeme prebacivanja

Pad medija stalne pohrane

- procedura za prebacivanje podataka na trajnu pohranu (slična stvaranju checkpoint-a)
  1. prebaci sve log zapise koji se nalaze u glavnoj memoriji na trajnu pohranu
  2. prebaci sve buffer blokove na disk
  3. kopiraj sadržaj baze podataka na trajnu pohranu
  4. zapiši log zapis <dump> na trajnu pohranu
- u slučaju da se dogodi djelomičan pad, samo se pogođeni blokovi mijenjaju
  - te se samo za njih provodi *redu* akcija

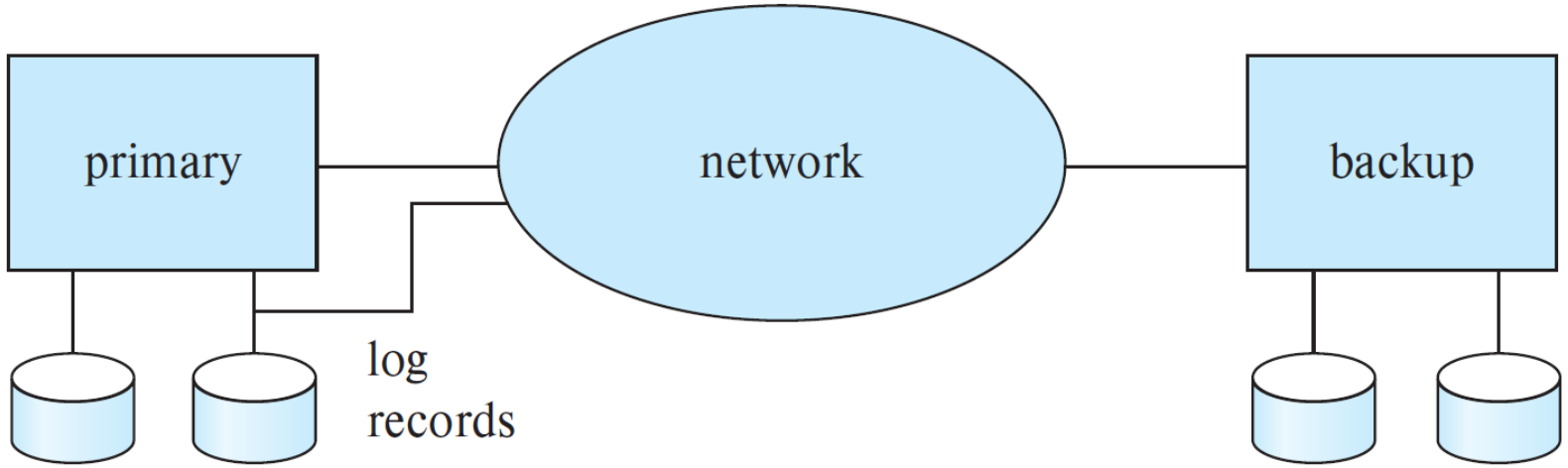
Udaljeni backup

- sustavi za obradu transakcija su u pravilu centralizirani ili client-server sustavi
  - što ih čini jako osjetljivima na prirodne katastrofe
- potreba za visokom dostupnosti
  - procesiranje transakcije na jednom (primarnom) mjestu (*engl. primary site*)
  - repliciranje podataka na drugo mjesto (*engl. secondary site*)
- sekundarna lokacija, udaljeni backup, mora biti sinkronizirana s prvom
  - sinkronizacija se postiže slanjem logova s primarne lokacije na sekundarnu

Udaljeni backup

- dvije lokacije trebaju biti fizički udaljene zbog prirodnih katastrofa: poplave, potresi…
  - idealno druga država, druga tektonska ploča…
- kada primarna lokacija padne, obradu transakcija preuzima sekundarna
  - sekundarna lokacija prvo izvodi oporavak, s potencijalno zastarjelom kopijom podataka i logovima
  - standardni algoritmi za oporavak se mogu koristiti
- kada se **dovrši oporavak** sekundarna lokacija preuzima obradu transakcija

Udaljeni backup



- arhitektura udaljenog backup sustava
- sustav se može oporaviti čak i ako su svi podaci s primarne lokacije izgubljeni
  - visoka dostupnost

Udaljeni backup

- nekoliko je izazova prilikom korištenja udaljenog backup-a:
  - detekcija kvara
    - važno je da udaljeni sustav otkrije kvar na primarnom
    - pad komunikacijske veze ne znači i pad primarnog sustava
    - rješenje: održavanje više nezavisnih veza između dvije lokacije
  - transfer kontrole
    - kada primarni sustav padne, sekundarni postoje novi primarni
    - kontrola se može transferirati ručno ili automatski (koristeći software DB vendora)
    - upiti se šalju na novi primarni sustav
      - mnogi sustavi dodijele IP adresu starog novom primarnom
      - korištenje proxy-a
    - jednom kada se stari primarni oporavi može služiti ponovno kao primarni ili backup

Udaljeni backup

- … još neki izazovi:
  - vrijeme za oporavak
    - ako na sekundarnom sustavu log naraste preveliki, oporavak može dugo trajati
    - sekundarni sustav može povremeno procesirati log-ove te stvoriti checkpoint
    - **hot-spare configuration** - redo logovi se procesiraju kako stižu, najkraće vrijeme zastoja
  - vrijeme za commit
    - transakcija ne smije biti proglašena commit-anom dok se log zapisi na evidentiraju na sekundarnoj lokaciji
    - stupnjevi trajnosti transakcije:
      - one-safe: upis na trajnu pohranu primarnog sustava
      - two-vary-safe: upis na trajnu pohranu primarnog i sekundarnog sustava
      - two-safe: isto kao i prethodna ukoliko su obje lokacije aktivne

Udaljeni backup

- većina sustava BP podržava udaljeni backup i opisane funkcionalnosti
  - backup se može izvršiti na više lokacija
- alternativni način za ostvarivanje visoke dostupnosti je korištenje **distribuiranih baza podataka**
  - podaci se repliciraju na više mjesta
  - ako je dobro implementirano mogu imati višu razinu dostupnosti od sustava udaljenog backup-a
- krajnji korisnici interakciju tipično ostvaruju preko aplikacija, a ne direktnim pristupom bazi
  - korištenje **load-balancer-a**

Literatura

- Pročitati
  - [DSC] poglavlje 16.1. – 16.6.
- Slijedeće predavanje
  - [DSC] poglavlje 17.
  - [DSC] poglavlje 18.
  - [DSC] poglavlje 19.