Kontrola konkurentnosti

BP 2 2022/23

TRANSAKCIJA

Transakcija je niz operacija nad bazom podataka koji odgovara jednoj logičkoj jedinici posla u realnom sistemu.

Move $100 from Andy' bank account to his promotor's account.

Transaction:
→ Check whether Andy has $100.
→ Deduct $100 from his account.
→ Add $100 to his promotor account.

IZVRŠAVANJE TRANSAKCIJA

- Pretpostavimo da u multiprogramskom računarskom sistemu postoji skup od n aktivnih transakcija,
  T={T1 ,…, Tn}.
- Transakcije iz ovog skupa se mogu izvršavati :
  - Serijski
  - Konkurentno (Neserijski) – uporedno izvršavanje

IZVRŠAVANJE TRANSAKCIJA

A = 100, B = 100

T_1	T_2
read(A)	
$A := A - 50$	
write (A)	
read(B)	
$B := B + 50$	
write(B)	
	read(A)
	$temp := A * 0.1$
	$A := A - temp$
	write(A)
	read(B)
	$B := B + temp$
	write(B)

A = 45, B = 155

T_1	T_2
	read(A)
	$temp := A * 0.1$
	$A := A - temp$
	write(A)
	read(B)
	$B := B + temp$
	write(B)
read(A)	
$A := A - 50$	
write(A)	
read(B)	
$B := B + 50$	
write(B)	

A = 40, B = 160

T_1	T_2
read(A)	
$A := A - 50$	
write(A) 50	
	read(A) 50
	$temp := A * 0.1$
	$A := A - temp$
	write(A) 45
read(B)	
$B := B + 50$	
write(B) 150	
	read(B) 150
	$B := B + temp$
	write(B) 155

A = 45, B = 155

T_1	T_2
read(A) 100	
$A := A - 50$	
	read(A) 100
	$temp := A * 0.1$
	$A := A - temp$
	write(A) 90
	read(B)
write(A) 50	
read(B) 100	
$B := B + 50$	
write(B) 150	
	$B := B + temp$
	write(B) 160

A = 50 B = 160

ACID

Transakcija u izvršenju mora da ima tzv. **ACID** osobine:

- **Atomičnost (Atomicity)**. Zahteva se da se sve operacije nad bazom podataka predviđene transakcijom uspešno obave ili da se ne prihvati nijedna, tj. ako se bar jedna operacija ne obavi kako je očekivano, onda se poništavaju efekti i svih ostalih.
- **Konzistentnost (Consistency)**. Pre početka i posle okončanja transakcije stanje baze podataka mora da zadovolji uslove konzistentnosti. Za vreme obavljanja transakcije konzistentnost baze podataka može da bude narušena.
- **Izolacija (Isolation)**. Kada se dve ili više transakcija izvršavaju istovremeno, njihovi efekti moraju biti međusobno izolovani. Drugim rečima efekti koje izazovu transakcije koje se obavljaju istovremeno moraju biti jednaki efektima nekog njihovog serijskog (jedna posle druge) izvršenja.
- **Trajnost (Durability)**. Kada se transakcija završi njeni efekti ne mogu biti izgubljeni, čak i ako se neposredno po njenom okončanju desi neki ozbiljan otkaz sistema.

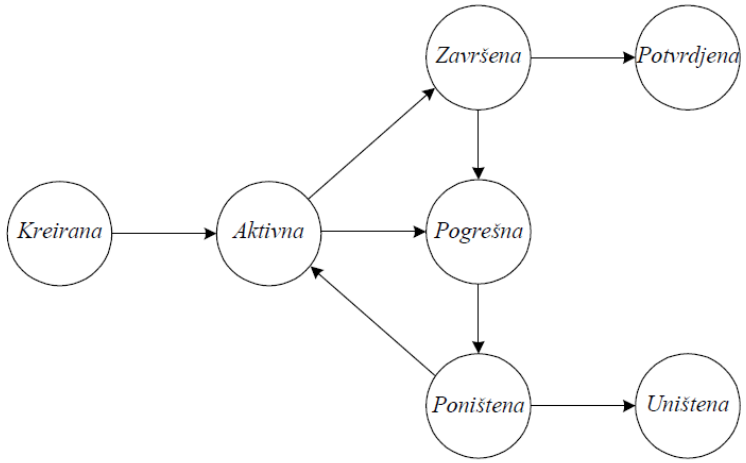
Atomicity: "all or nothing"

Consistency: "it looks correct to me"

Isolation: "as if alone"

Durability: "survive failures"

Stanja transakcije



- **Kreirana** – spremna za aktiviranje
- **Aktivna** – počne da se izvršava
- **Završena** – pošto se izvrši zadnja instrukcija.
- **Pogrešna** – ako se inicirani upisi u bazu ne završe usled kvara sistema. U ovo stanje, transakcija može preći i iz aktivnog stanja, ukoliko u toku izvršavanja transakcije dođe do kvara sistema, bilo hardverskog, bilo softverskog.
- **Poništena** – Pošto se nije uspešno završila, transakciju je potrebno prevesti u stanje Poništena, i vratiti bazu u prethodno konzistentno stanje. Posle restauracije baze, transakciju treba ponovo startovati, ako je hardverski ili softverski kvar, zbog koga se transakcija nije mogla završiti, u potpunosti otklonjen. Ukoliko kvar ne može biti u potpunosti otklonjen, ponovno startovanje transakcije nema smisla, već se transakcija prevodi u stanje Uništena.
- **Uništena** – Tipičan slučaj je prisustvo logičke greške u transakciji. Takva greška može biti ispravljena samo ponovnim pisanjem programa transakcije.
- **Potvrđena** – svi upisi, inicirani u toku izvršenja transakcije, u bazu završeni i očuvana konzistentnost baze.

Izolacija

Konkurentno izvršavanje

- Rezultat konkurentnog izvršavanja transakcija mora biti ekvivalentan razultatu koji se dobija serijskim izvođenjem tih istih transakcija.
- Neki neserijski redosled je serijabilan (linearan) ako je ekvivalentan serijskom redosledu.
- Osnovna Pretpostavka – Bez obzira na način izvršavanja, po završetku kompletne transakcije stanje baze mora biti **konzistetno**.

KONKURENTNOST - problemi

Nekontrolisano konkurentno izvršenje transakcija može uzrokovati:

- Gubljenje rezultata ažuriranja – pisanje preko ažuriranih podataka
- Problem nekorektne analize podataka – upotreba pre ažuriranja
- Problem nepotvrđenih promena (čitanja)

Mehanizam čuvanja izolovanosti

Protokol kontrole konkurentnosti - način odlučivanja o preplitanju operacija više transakcija.

Dve kategorije protokola:

- Pesimistični: Ne dozvoljavaju da se problemi uopšte pojave.
- Optimistični: Pretpostavlja se da su konflikti retki, pa se definišu mehanizmi za njihovo rešavanje kada do istih dođe.

Rasporedi izvršavanja

Dva rasporeda izvršavanja su ekvivalentna ako je efekat izvršavanja jednog identičan efaktu izvršavanja drugog, za bilo koge konzistentno stanje baze.

Da li je određeni raspored izvršavanja više transakcija korektan ili nije se odlučuje na osnovu toga

da li je ekvivalentan nekom serijskom rasporedu.

Rezultat konkurentnog izvršavanja transakcija mora biti ekvivalentan razultatu koji se dobija serijskim izvođenjem tih istih transakcija.

Konfliktne operacije

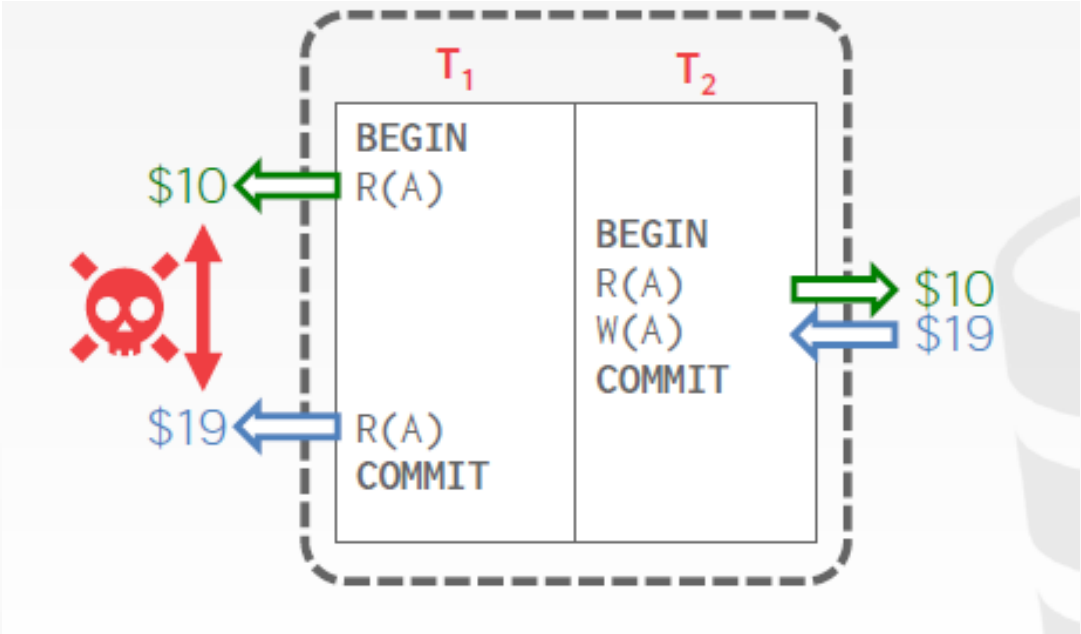
Dve operacije su **konfliktne** ako:

- Pripadaju različitim transakcijama
- Rade sa istim objektom, pri čemu bar jedna od te dve operacije pisanje.

Read-Write Conflicts (**R**-**W**)
Write-Read Conflicts (**W**-**R**)
Write-Write Conflicts (**W**-**W**)

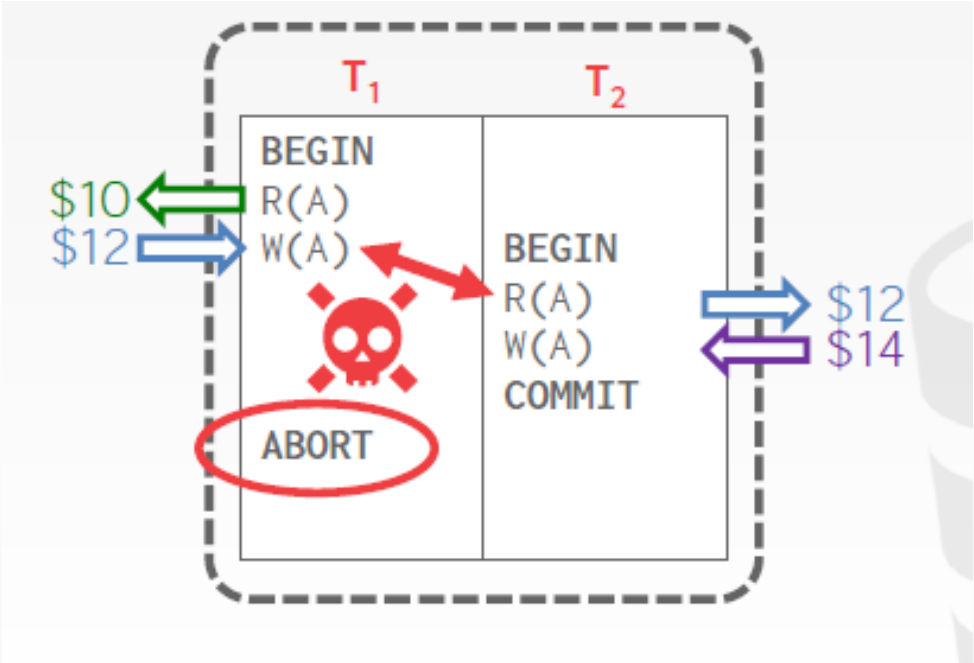
READ-WRITE CONFLICTS

Neponovljivo čitanje



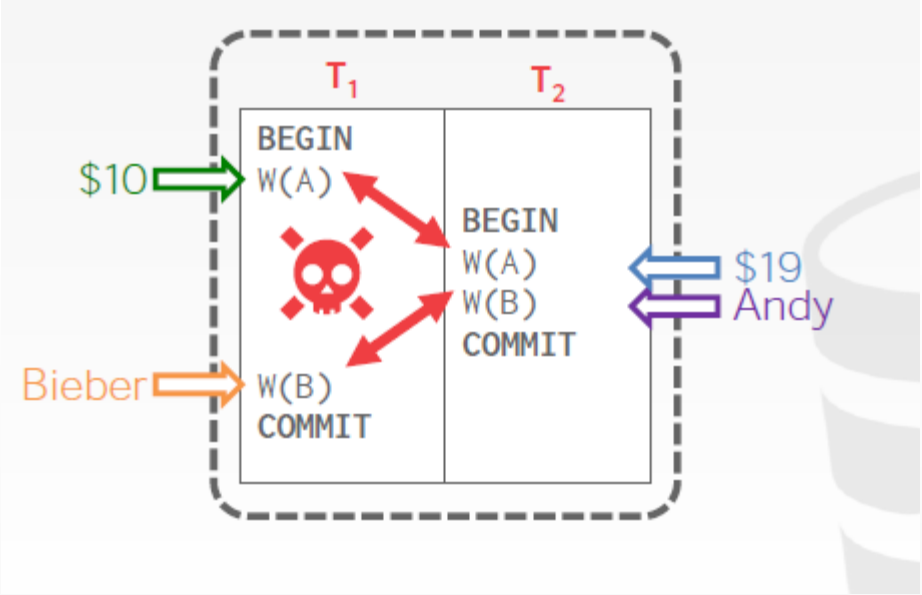
WRITE-READ CONFLICTS

Prljavo čitanje (čitanje nepotvrđenih podataka)



WRITE-WRITE CONFLICTS

Prepisivanje nepotvrđenih podataka (prljavo pisanje)

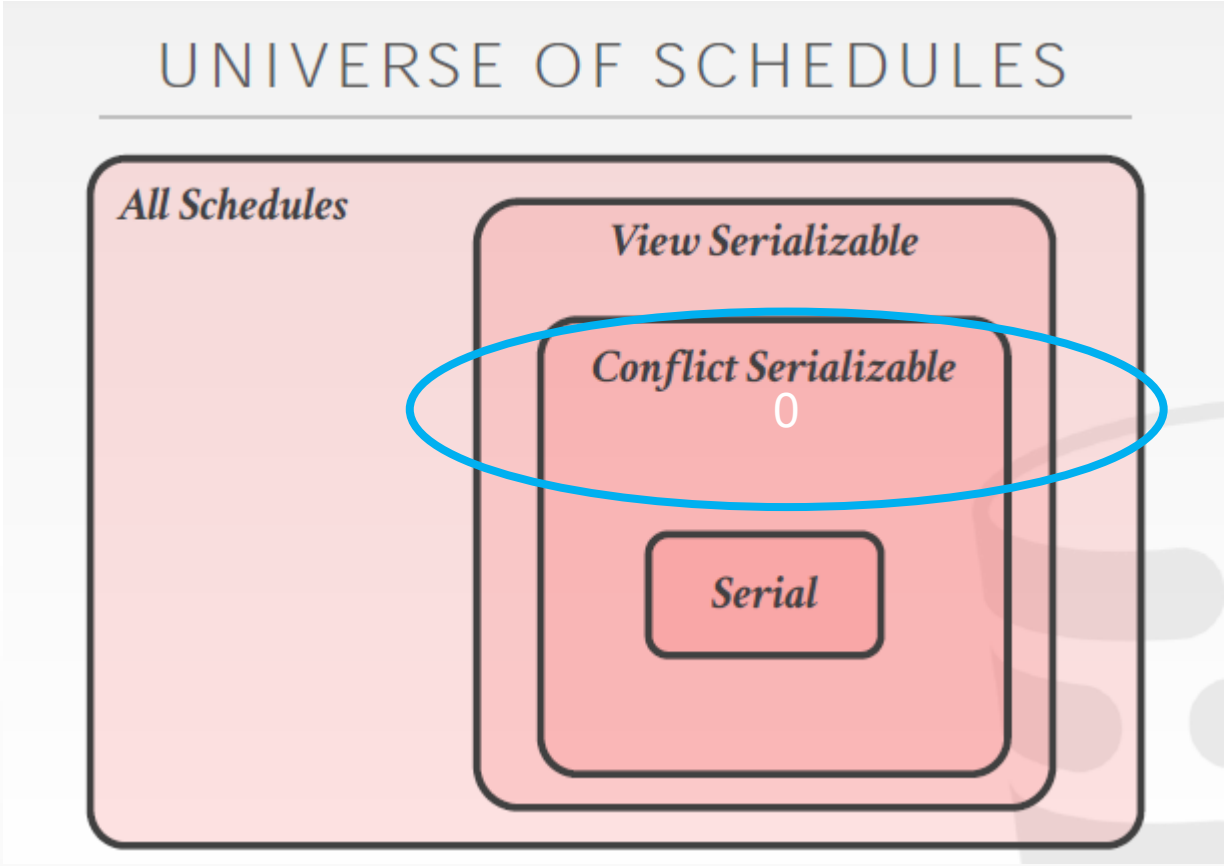


VRSTE RASPOREDA

Neserijski raspored može serijazovan. Postavlja se pitanje kako utvrditi da li je serijazovan.

Postoje dva nivoa serjazovanosti:

- Konfliktna – većina DMBSova podržava ovu vrstu provere pri definisanju rasporeda
- View serijalizovanost



KONFLIKTNA SERIJALIZOVANOST

Dva rasporeda su konfliktno ekvivalentna akko:

- Uključuju iste akcije istih transakcija, i
- Svaki par konfliktnih radnji je raspoređen na isti način.

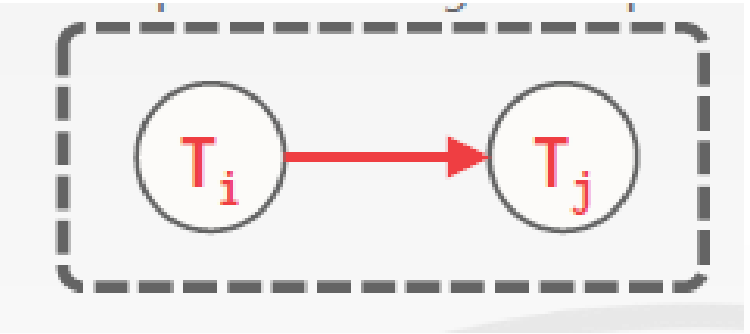
Raspored je konfliktno serijalizovan ako je konfliktno ekvivalentan nekom serijskom rasporedu.

KAKO UTVRDITI KONFLIKTNU SERIJALIZOVANOST

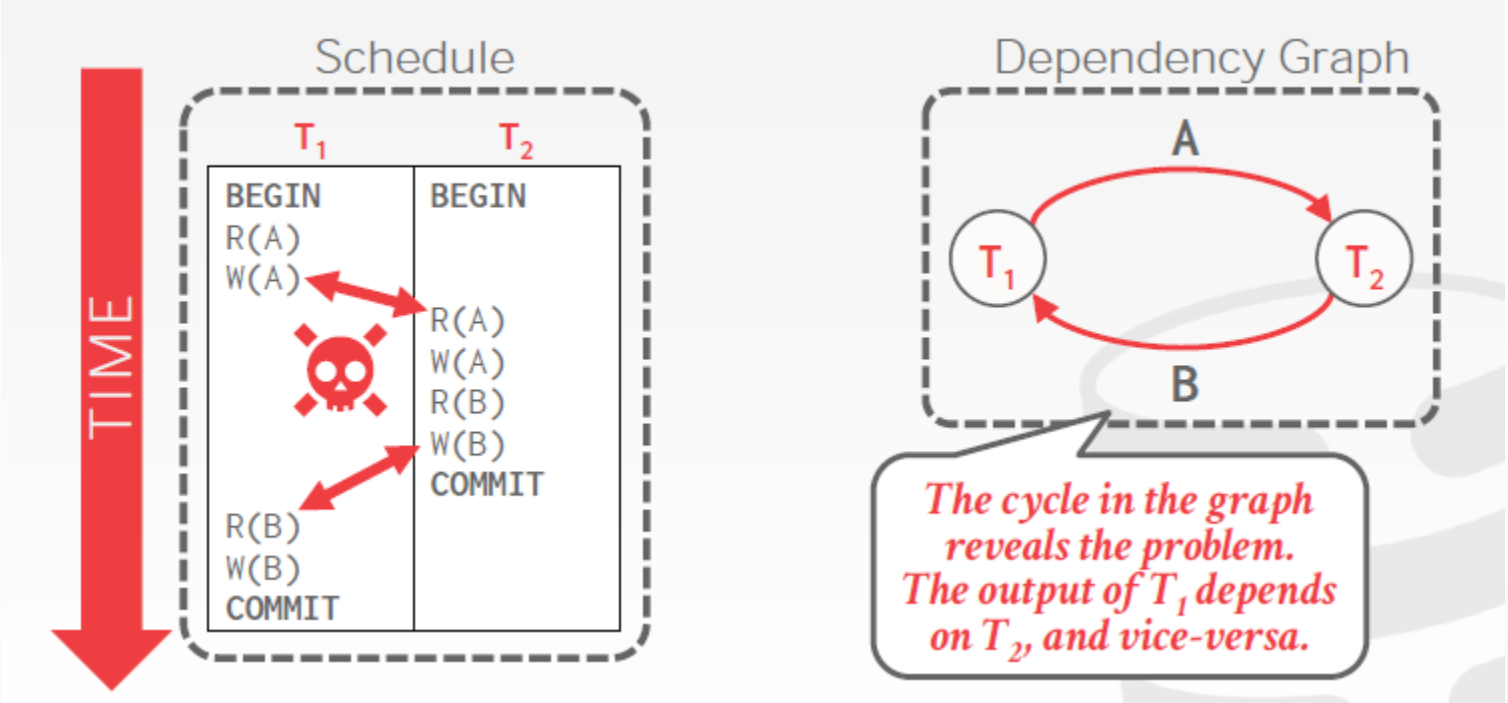
Definisanjem grafa prioriteta, tj. grafa zavisnosti.
Čvorovi grafa su transakcije. Od Ti do Tj postoji grana ako:

- Operacija Oi transakcije Ti je u konfliktu sa operacijom Oj transakcije Tj i
- Oi je ranije u rasporedu od Oj.

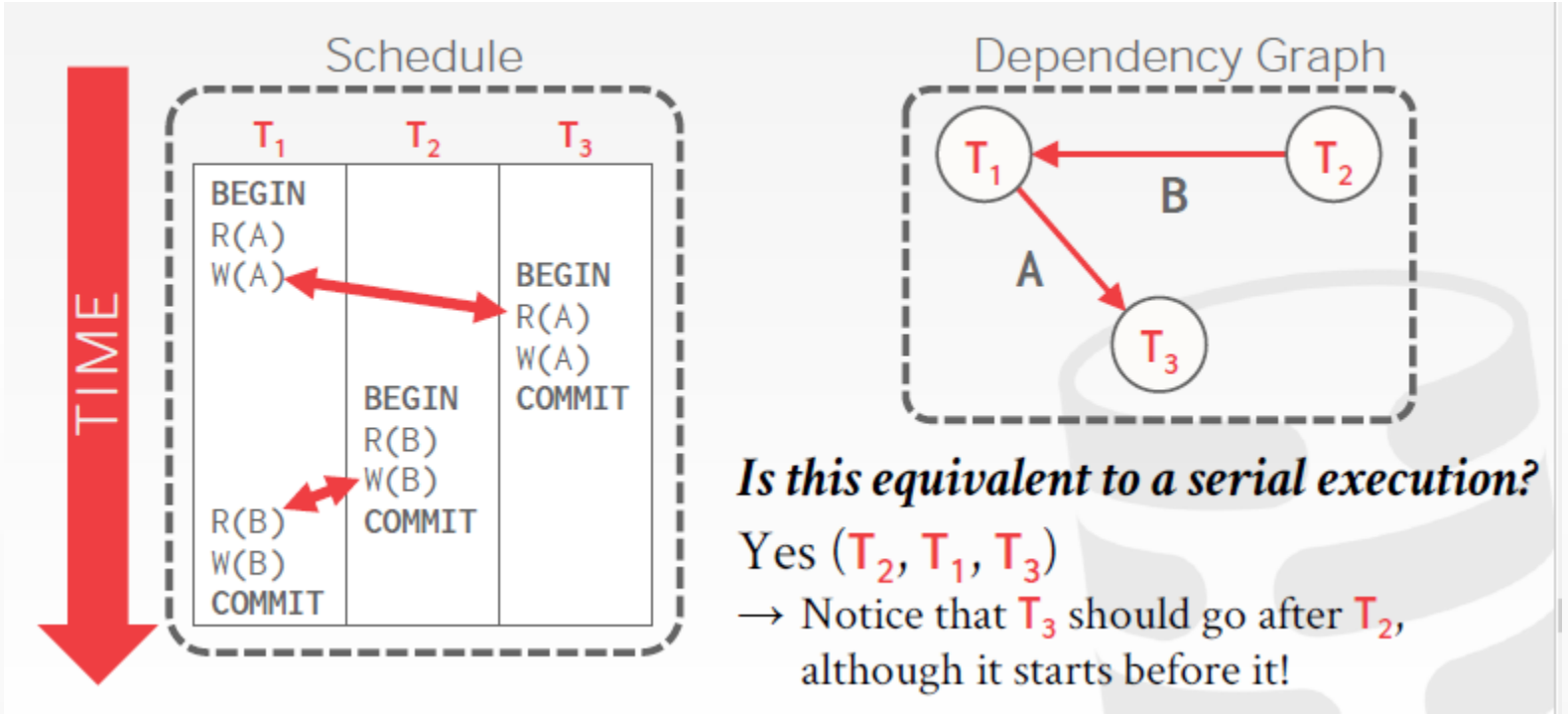
Raspored je konfliktno serijalizovan akko je njegov graf zavisosti acikličan.



GRAF ZAVISNOSTI

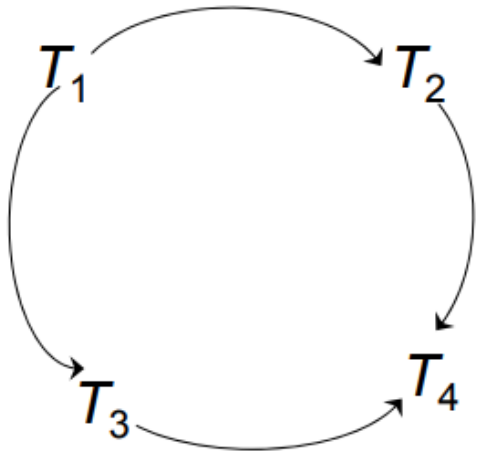


GRAF ZAVISNOSTI



PRIMER

T_1	T_2	T_3	T_4	T_5
	read(X)			
read(Y)				
read(Z)				
				read(V)
				read(W)
				read(W)
	read(Y)			
	write(Y)			
		write(Z)		
read(U)				
			read(Y)	
			write(Y)	
			read(Z)	
			write(Z)	
read(U)				
write(U)				

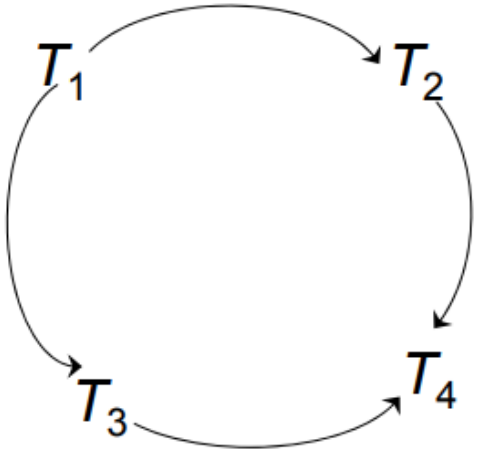


PRIMER

Ako je graf prioriteta acikličan, redosled seriabilnosti može se dobiti pomoću topološkog sortiranja grafa.

Na primer,
T5 → T1 → T3 → T2 → T4

*Topološko sortiranje DAGa je linearno raspoređivanje svih njegovih čvolorva tako da ako graf sadrži vezu (u,v), onda se u nizu čvor u javlja pre čvora v.



PROTOKOLI ZAKLJUČAVANJA

KONTROLA SERIJALIZOVANOSTI

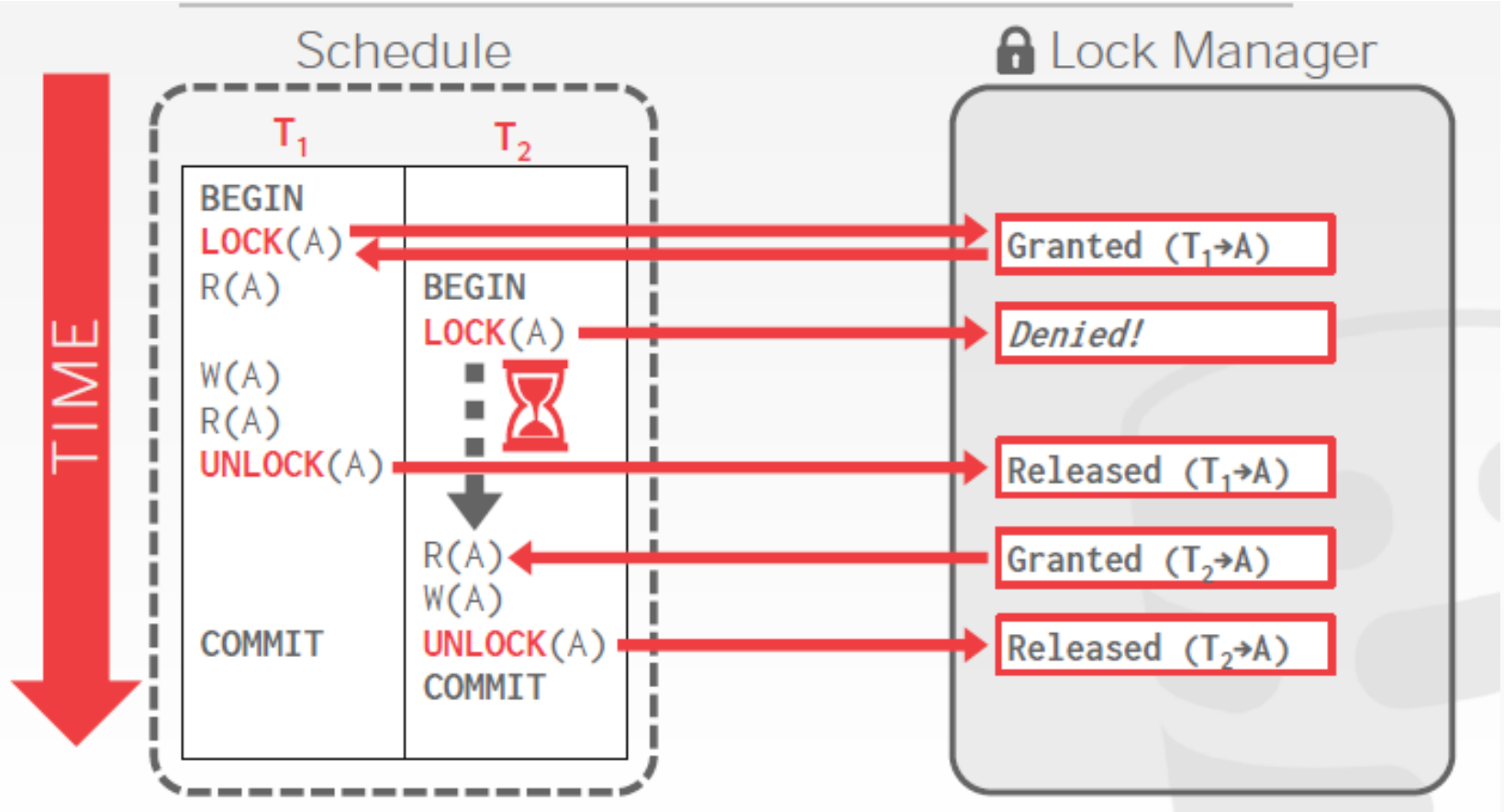
DBMS mora da ima mehanizam koji će da obezbediti da svi rasporedi budu
- Serijabilni i
- Da se mogu oporaviti u slučaju otkaza

Cilj – razviti protokole za kontrolu konkurentnosti koji će obezbediti serijalizovanost rasporeda.

KONTROLA SERIJALIZOVANOSTI

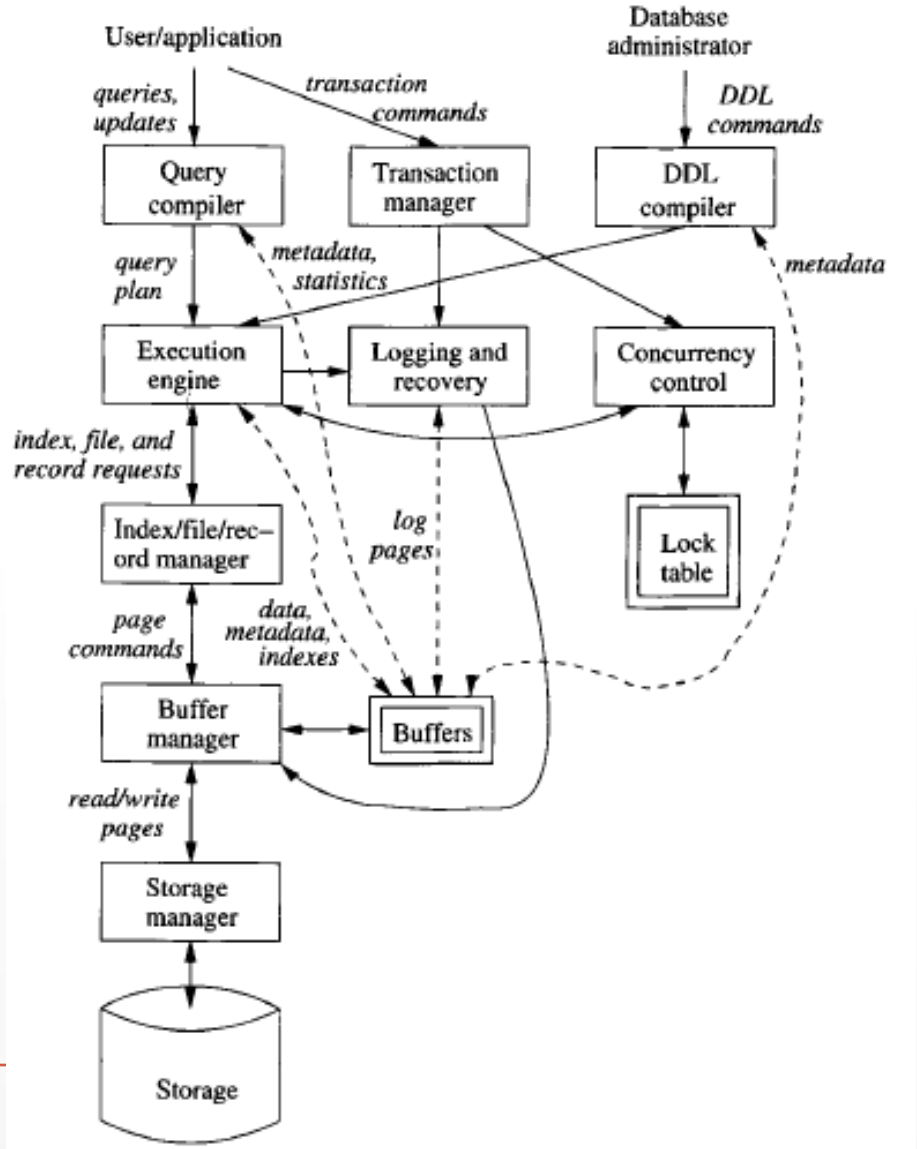
- Protokoli za kontrolu konkurentnosti – dozvoljavaju neserijske rasporede, ali obezbeđuju serijabilnost i oporavak.
- Protokoli za kontrolu konkurentnosti generalno **ne ispituju graf prioriteta** već nameću disciplinu kojom se izbegavaju nepoželjni rasporedi.
- Testovi serijalizovanosti (kao DAG) služe tome da se pokaže da je neki protokol korektan.
- Vrste protokola:
  - Protokoli bazirani na zaključavanju
  - Protokoli bazirani na grafovima
  - Protokol vremenskog markiranja

Protokoli bazirani na zaključavanju

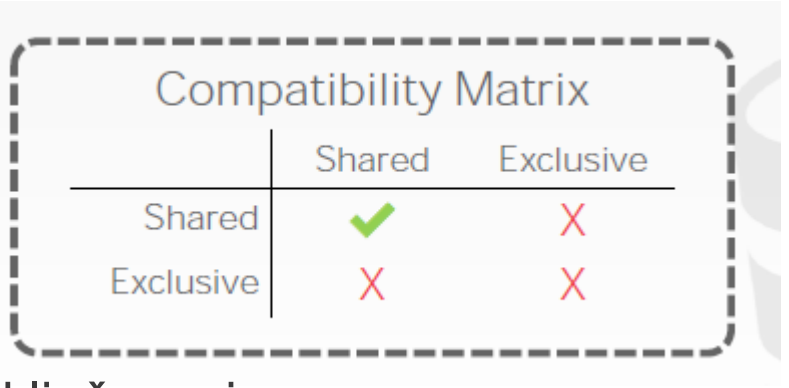


DBMS komponenta

- Transaction Manager - DBMS poseduje komponentu koja upravlja celokupnim izvršenjem transakcija.
- Kontrolu izvršavanja konkurentnih trasakcija vrši Concurency control komponenta, tj. planer (Scheduler).
- Zadatak planera – sprečava neserijabilna rešenja primenom nekog od protokola za utvrđivanje i/ili obezbeđivanje serijabilnosti.
- Protokolima se obezbedjuje serijalizovanost i to NAMETANJEM DISCIPLINE kojom se izbegavaju neserijabilni scenariji izvođenja.

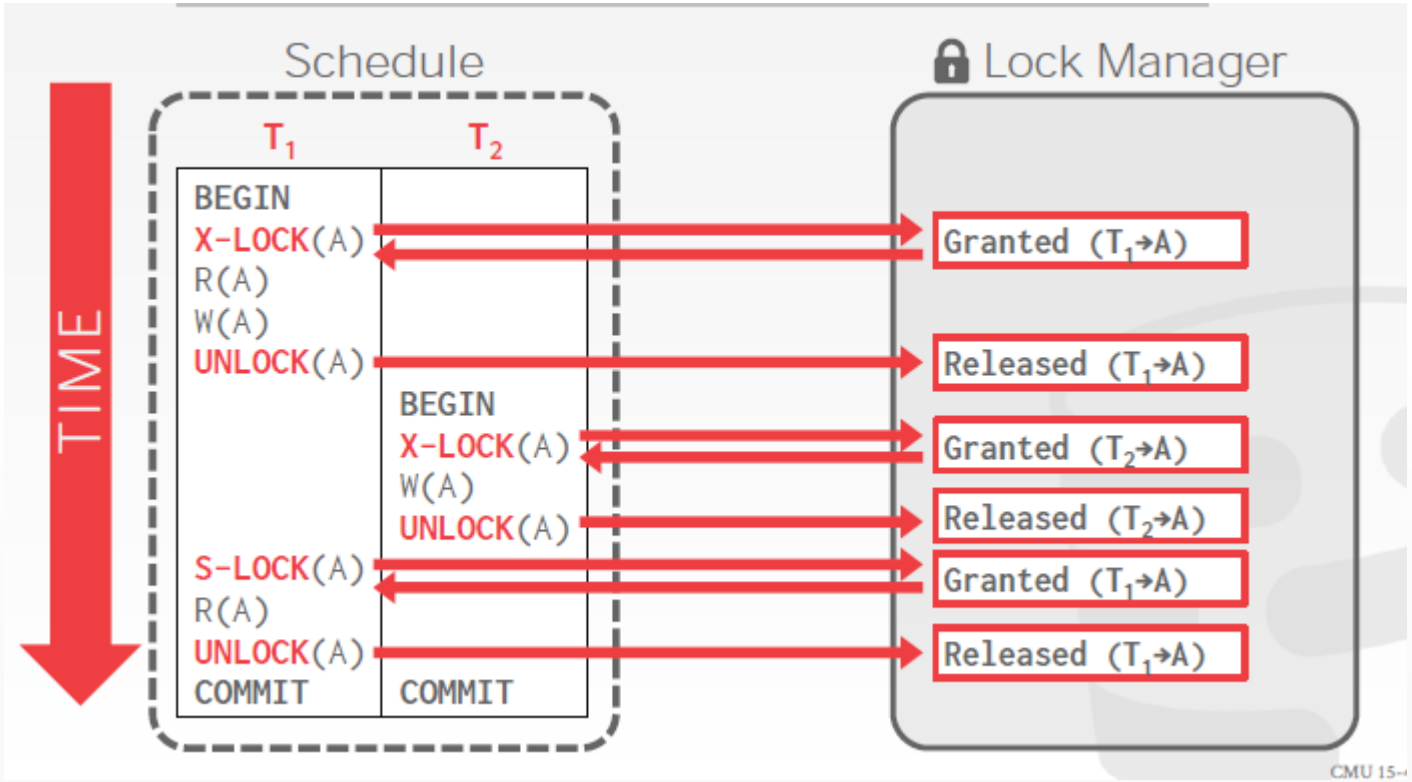


Protokol zaključavanja

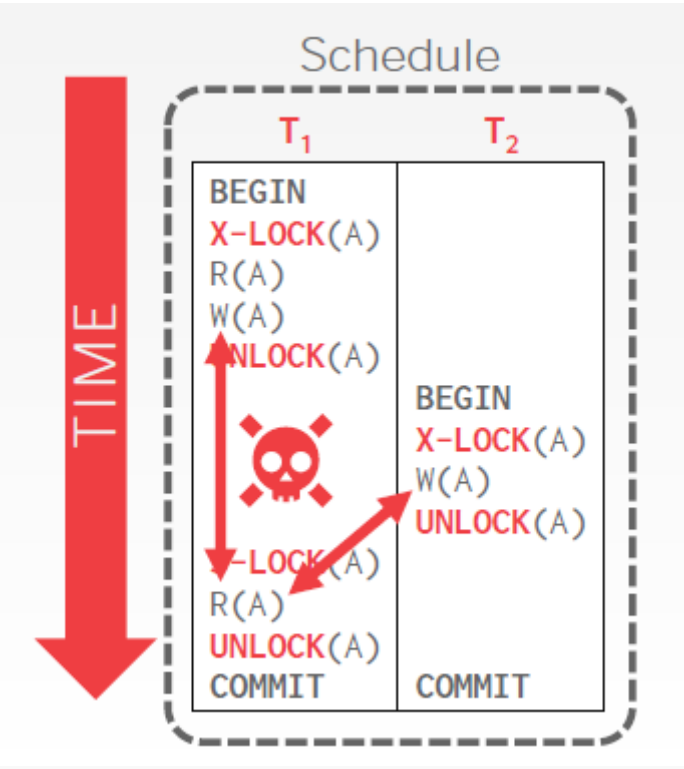


- Forisarano ostvarivanje serijabilnosti primenom mehanizma zaključavanja.
- Tokom izvršenja transakcije se postavlja lokot (lock) na objekat baze kojem je pristupila.
- Osnovne vrste zaključavanja podataka
  - **Ekskluzivno zaključavanje** (**exclusive** ili write lock) – ni jedna druga transakcija ne može postaviti novi lokot, bilo koje vrste.
  - **Deljivo zaključavanje** (**shared** (S) ili read lock) – druge transakcije mogu dobiti deljivi lock nad istim objektom, ali ne mogu dobiti ekskluzivni.
- Zahtevi za zaključavanjem se prosleđuju concurrency-control manageru, a transakcija može nastaviti sa izvršavanjem samo ako je lokot odobren.

Primer



Neponovljivo čitanje



Problem sa S/X lokovima

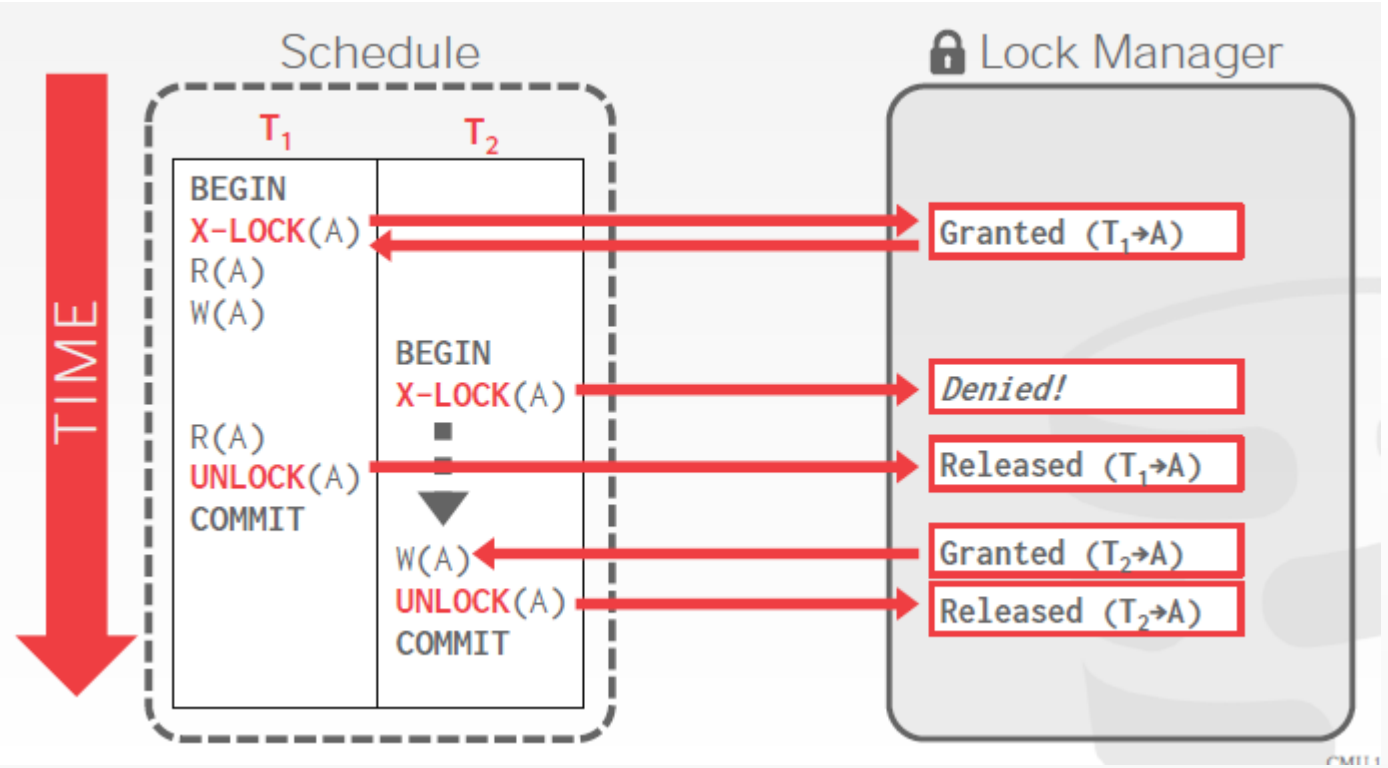
T_2: **lock-S**(*A*);
read (*A*);
unlock(*A*);
lock-S(*B*);
read (*B*);
unlock(*B*);
display(*A*+*B*)

Problem:
Ako su A i/ili B ažurirani između čitanja A i B, prikazani zbir bi bio pogrešan.

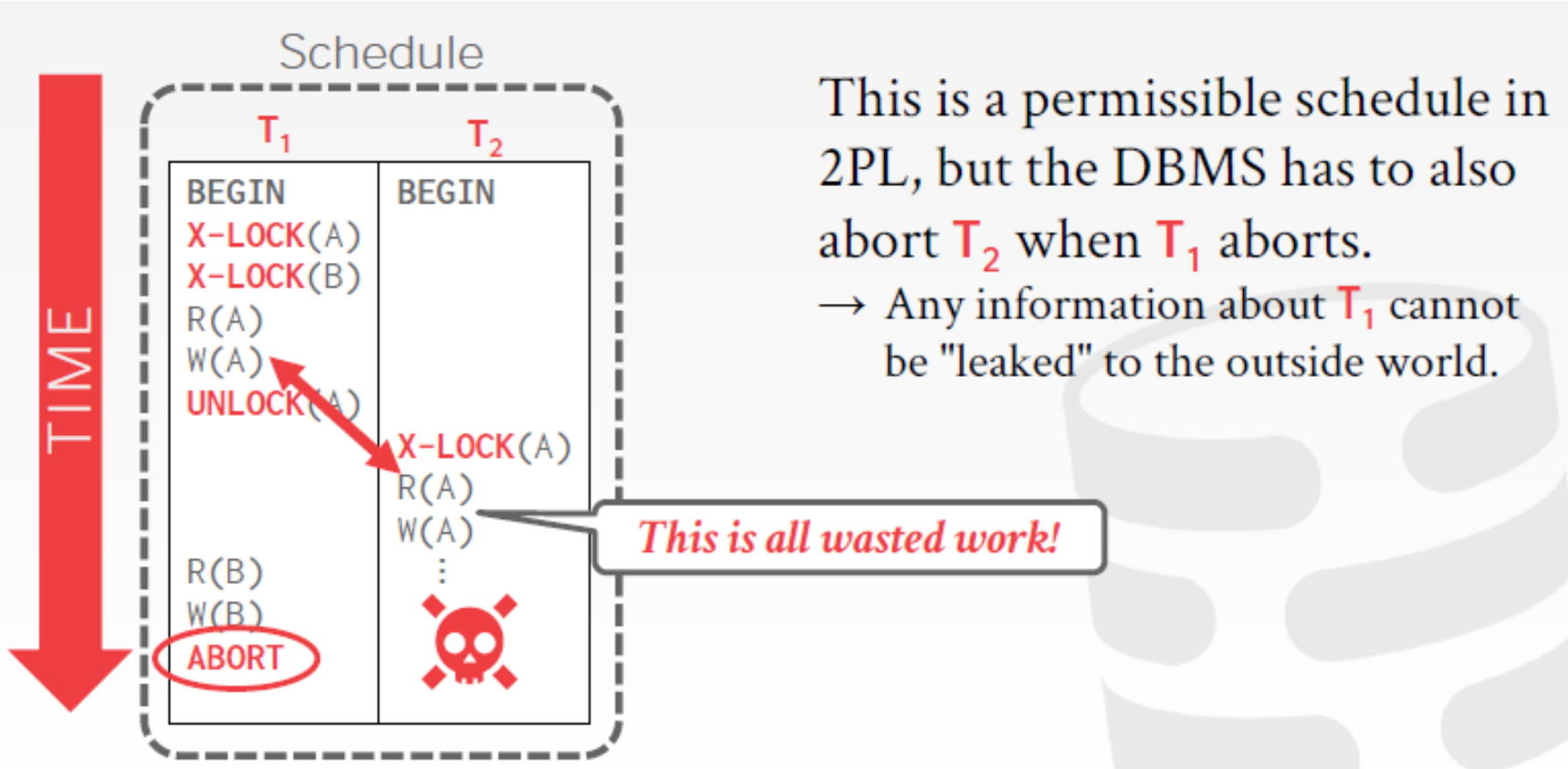
Dakle, nije dovoljno samo obezbediti S,X lokove.

Rešenje - Dvofazni protokol zaključavanja

- Transakcija prolazi kroz dve faze izvršavanja:
  - Faza širenja – kada broj lokota raste, lokoti se zahtevaju, ali se ne otpuštaju.
  - Faza skupljanja – lokoti se oslobađaju, a nije dozvoljeno postavljanje novih.
- Transakcije koje poštuju dvofazni protokol su uvek serijabilne.



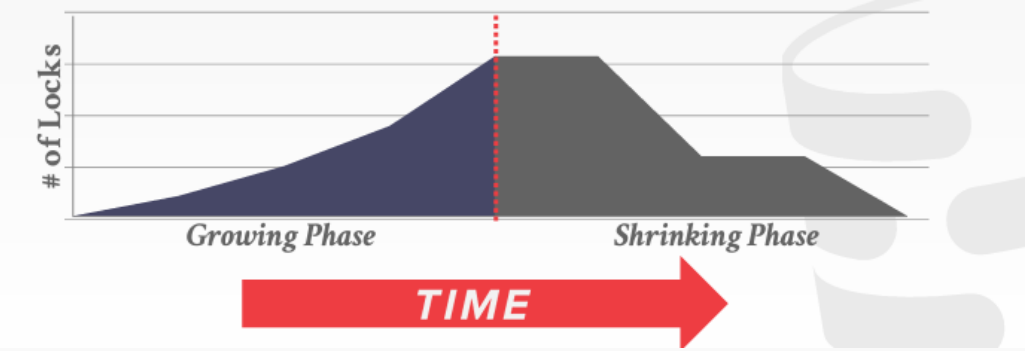
Dvofazni protokol zaključavanja – kaskadni otkaz



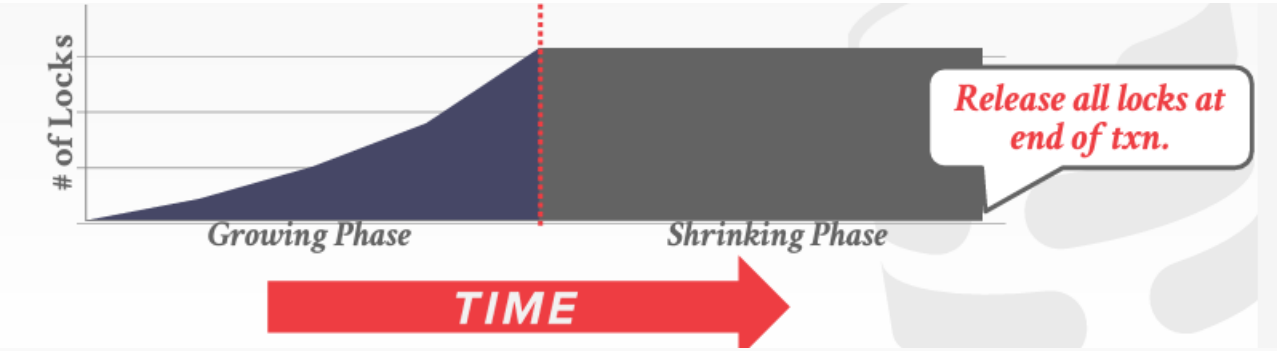
Rigorozni dvofazni protokol zaključavanja

Vrednost upisana od strane jedne transakcije ne može biti pročitana ili prepisana dok se transakcija ne završi.

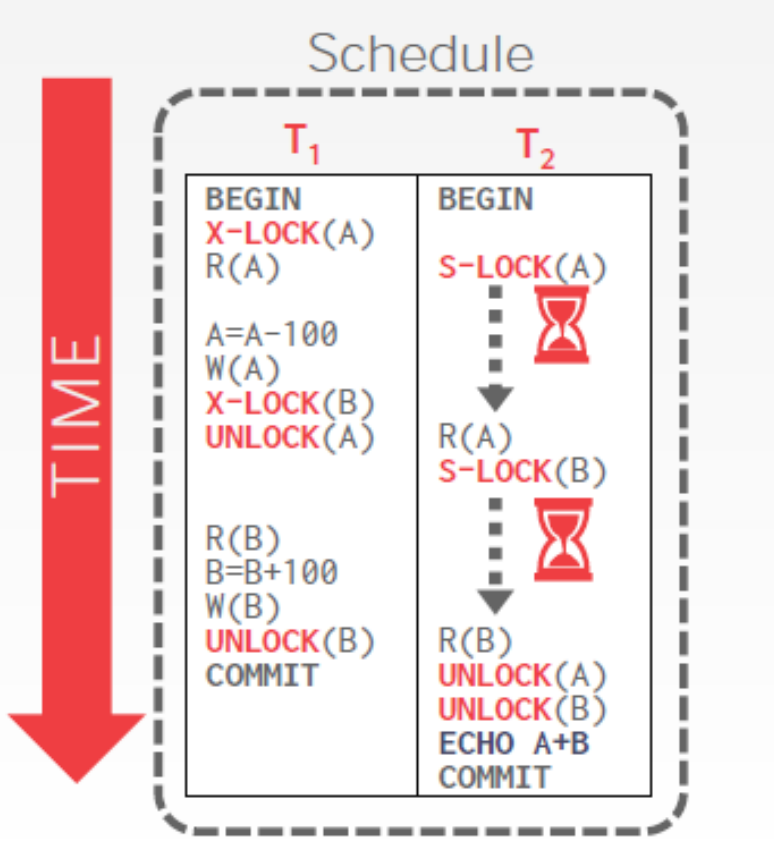
Dvofazni protokol

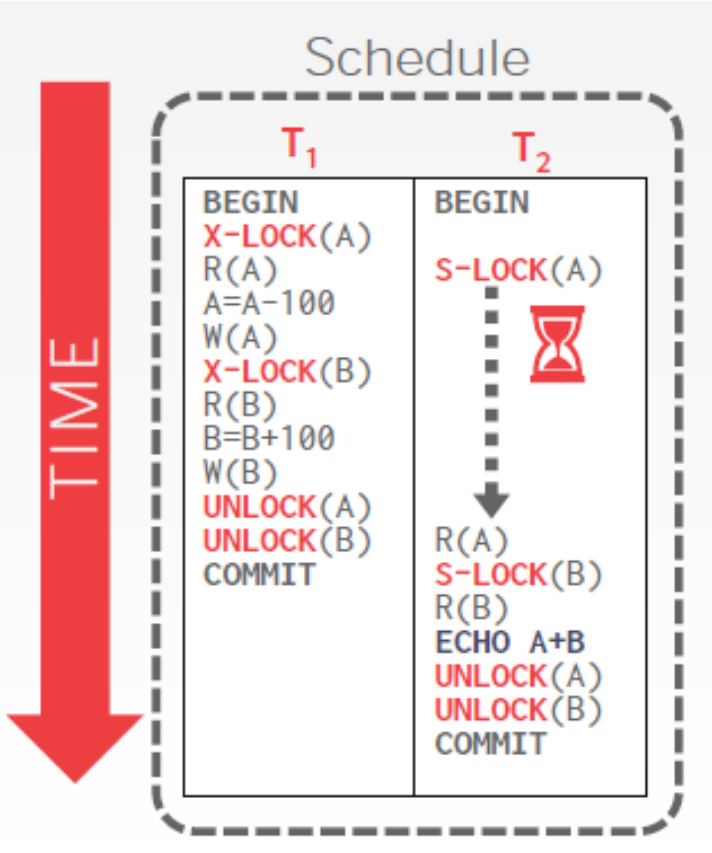


Rigorozni dvofazni protokol



Rigorozni dvofazni protokol zaključavanja

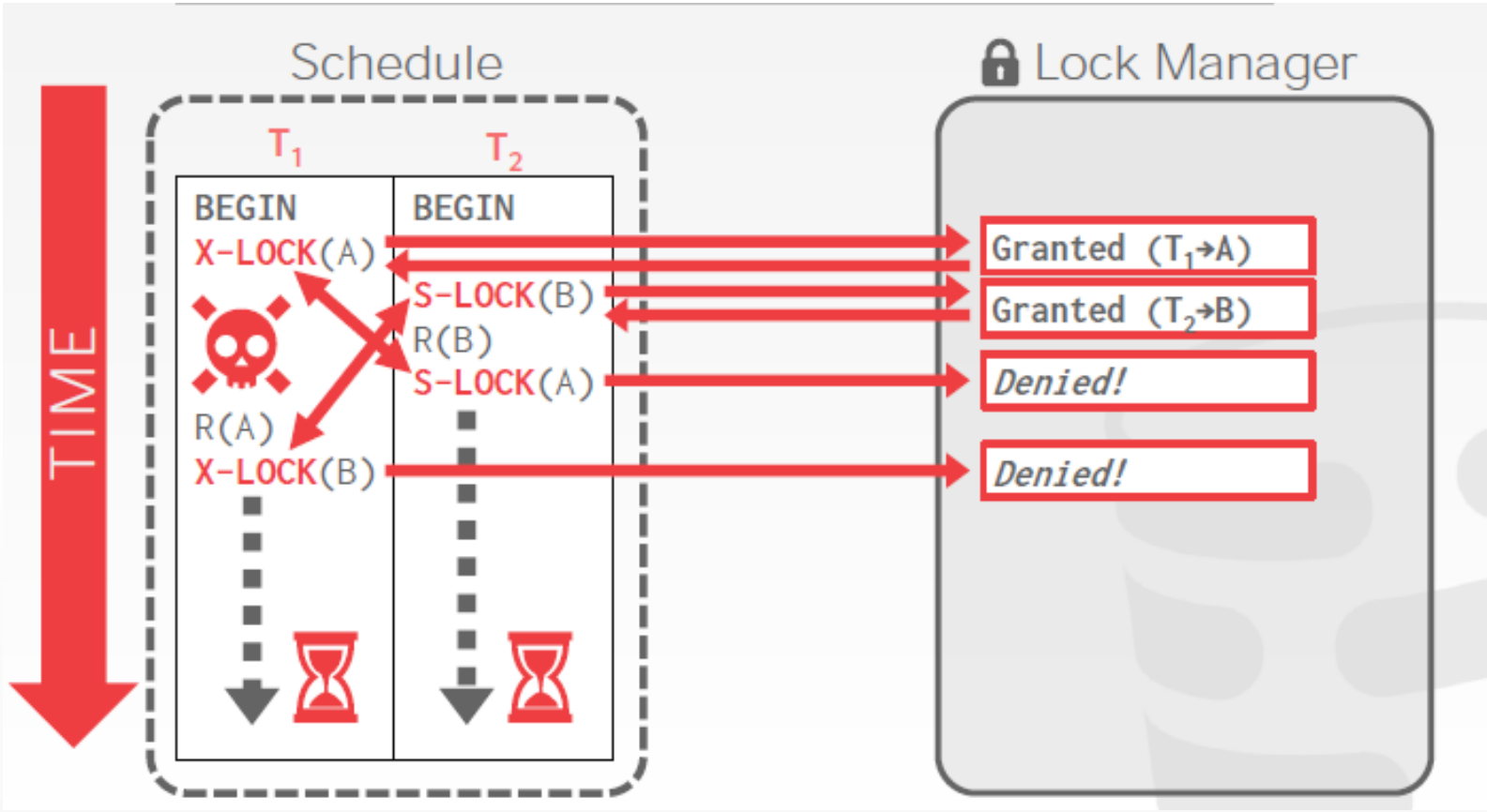




MRTVI ČVOR (izgladnjivanje)

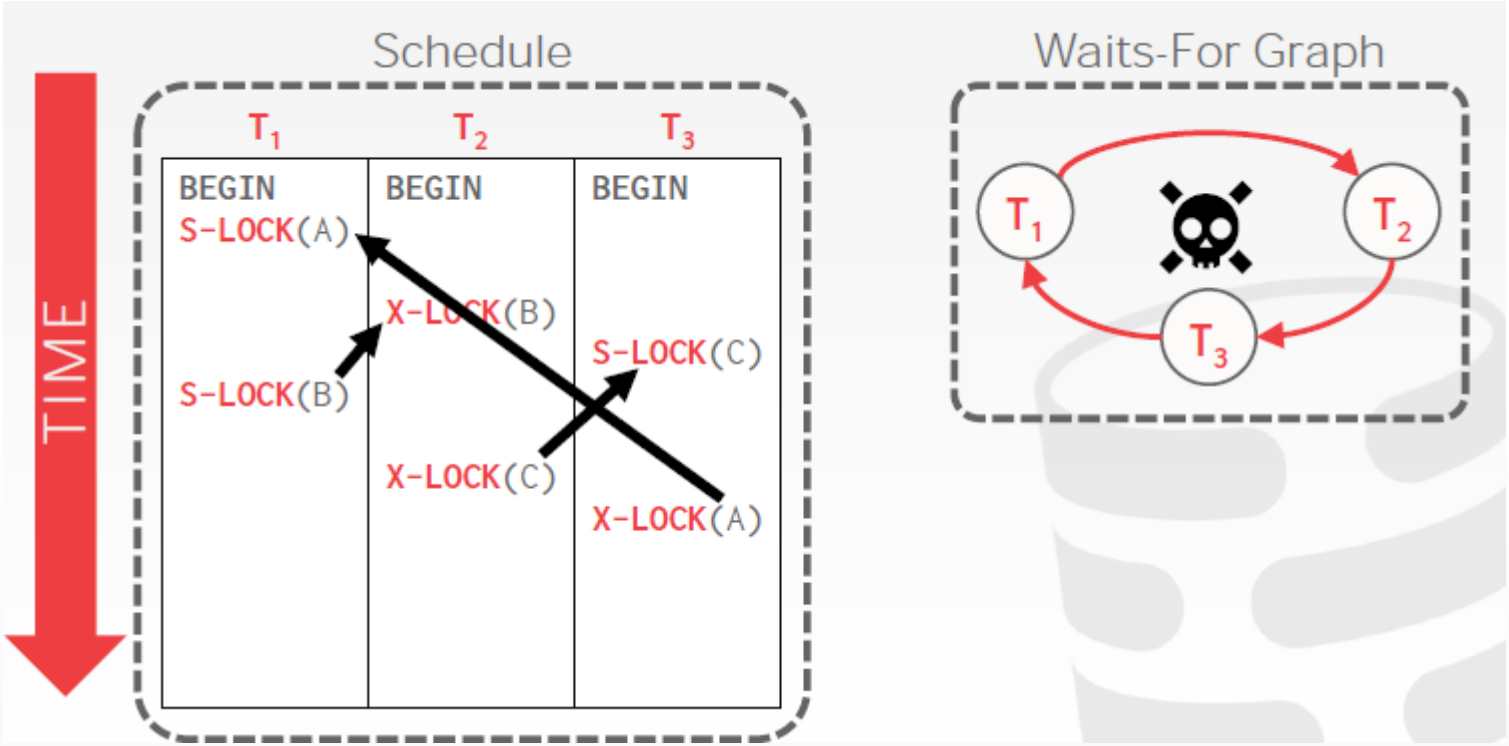
Mrtvi čvor

Situacija u koj dve transakcije čekaju jedna drugu na oslobađanje lokota.



Detekcija mrtvog čvora

DBMS kreira graf čekanja. Povremeno proverava ima li ciklusa u grafu i ako ga ima pravi odluku kako da ga prekine.



Sanacija mrtvog čvora

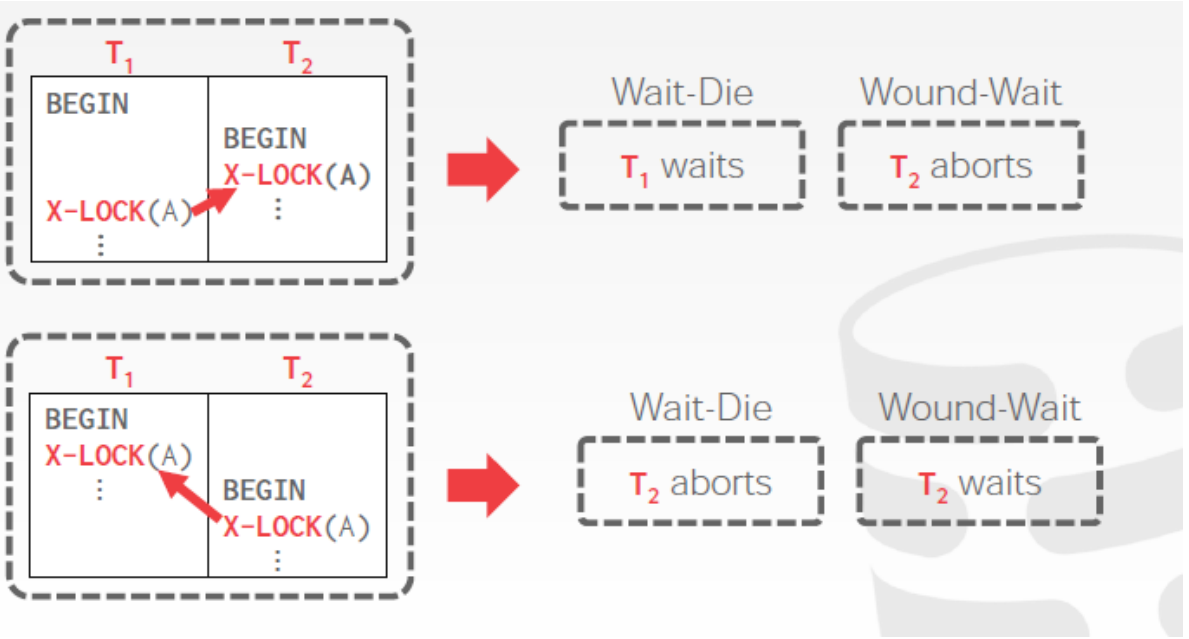
DBMS bira krivca koji će biti poništen.
Bira na osnovu:

- Starosti transakcije
- Broja izvršenih operacija
- Broja objekata koje je zaključala
- Broja transakcija koje također moraju biti poništene

Transakcija može biti poništena u potpunosti ili minimalno. Zavisi od politike DMBSa.

Prevencija mrtvog čvora

Odredi prioritete na osnovu vremena otpočinjanja.
Stariji vremenski pečat = Veći prioritet



Stari čekaju na mlade

- Ako je potražilac (lokota) stariji od vlasnika, onda se ostavlja na čekanju
- U suprotnom se poništava

Mladi čekaju na stare

- Ako je potražilac stariji od vlasnika, tada se vlasnik poništava i oslobađaju se lokoti
- U suprotnom biva poništena

Wait-Die ("Old Waits for Young")

→ If *requesting txn* has higher priority than *holding txn*, then *requesting txn* waits for *holding txn*.
→ Otherwise *requesting txn* aborts.

Wound-Wait ("Young Waits for Old")

→ If *requesting txn* has higher priority than *holding txn*, then *holding txn* aborts and releases lock.
→ Otherwise *requesting txn* waits.