



Dvanaesta Nedelja

– Transakcije: Obezbeđenje serijalizovanosti –

Autori: Miloš Cvetanović

- **Mehanizam Zaključavanja**
 - Binarno zaključavanje – $L(X)$, $UN(X)$
 - Kompleksno zaključavanje – $LS(X)$, $LX(X)$, $UN(X)$, (opciono: $DN(X)$, $UP(X)$)
- **Matrica kompatibilnosti za tipove zaključavanja**
- **Pravilno formirana transakcija**
- **Legalno izvršenje transakcija**

- **Dvofazni protokol zaključavanja**
 - Osnovni Dvofazni protokol zaključavanja (binarno zaključavanje)
 - Osnovni Dvofazni protokol zaključavanja (kompleksno zaključavanje)
 - Striktni Dvofazni protokol zaključavanja
- **Protokol u obliku stabla**
- **Mehanizam Vremenskog Markiranja**
 - Osnovni protokol Vremenskog Markiranja
 - Modifikovani protokol Vremenskog Markiranja
 - Striktni protokol Vremenskog Markiranja

Primer 5 – Mehanizam Vremenskog Markiranja (1)

- Na slici je prikazan redosled izvršavanja skupa transakcija T_1 , T_2 , T_3 , T_4 i T_5 .
- Upotrebom *modifikovanog protokola Vremenskog markiranja* proveriti da li se dobija isti redosled. U slučaju restartovanja transakcije odabrati najpovoljniji slučaj.
 - ako su vrednosti vremenskih marki:
 $TS(T_5) < TS(T_4) < TS(T_2) < TS(T_3) < TS(T_1)$.
 - ako su vrednosti vremenskih marki: (za vežbu)
 $TS(T_5) < TS(T_4) < TS(T_2) < TS(T_1) < TS(T_3)$.

Primer 5 – Mehanizam Vremenskog Markiranja (2)

Vreme	T_1	T_2	T_3	T_4	T_5
t_1					Read (E)
t_2					$E := E - 1$
t_3					Write (E)
t_4					Read (H)
t_5					$H := H - 1$
t_6					Write (H)
t_7					Commit
t_8				Read (A)	
t_9				$A := A - 1$	
t_{10}				Write (A)	
t_{11}		Read (B)			
t_{12}		Read (A)			
t_{13}		$B := B - 1$			
t_{14}		$A := B / 2$			
t_{15}		Read (G)			
t_{16}			Read (C)		
t_{17}			$C := C + 1$		
t_{18}			Write (C)		
t_{19}	Read (D)				
t_{20}	$D := D / 5$				
t_{21}	Read (C)				
t_{22}	$C := D * 2$				
t_{23}	Write (C)				
t_{24}	Write (D)				
t_{25}			Read (H)		
T_{26}			$H := H - 1$		

t_{27}			Write (H)		
t_{28}			Commit		
t_{29}	Read (I)				
t_{30}	$I := I - 1$				
t_{31}	Write (I)				
t_{32}	Commit				
t_{33}		Write (A)			
t_{34}		$G := G + A$			
t_{35}		Write (G)			
t_{36}				Read (E)	
t_{37}				$E := E / 3$	
t_{38}				Write (E)	
t_{39}				Commit	



Primer 5 – Mehanizam Vremenskog Markiranja (3)

T	Op	S	RA	WA	RB	WB	RC	WC	RD	WD	RE	WE	RG	WG	RH	WH	RI	WI	T1	T2	T3	T4	T5
			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	500	300	400	200	100
T5	READ (E)	OK									100												
T5	WRITE (E)	OK										100											
T5	READ (H)	OK													100								
T5	WRITE (H)	OK														100							
T5	COMMIT	OK																					COMMIT
T4	READ (A)	OK	200																				
T4	WRITE (A)	OK		200																			
T2	READ (B)	OK			300																		
T2	READ (A)	OK	300																				
T2	READ (G)	OK											300										
T3	READ (C)	OK					400																
T3	WRITE (C)	OK						400															
T1	READ (D)	OK							500														
T1	READ (C)	OK					500																
T1	WRITE (C)	OK						500															
T1	WRITE (D)	OK								500													
T3	READ (H)	OK													400								
T3	WRITE (H)	OK														400							
T3	COMMIT	?																			CF		
T1	READ (I)	OK															500						
T1	WRITE (I)	OK																500					
T1	COMMIT	?																	CF				
T2	WRITE (A)	OK	300																				
T2	WRITE (G)	OK											300										
T4	READ (E)	OK									200												
T4	WRITE (E)	OK										200											
T4	COMMIT	OK																				COMMIT	

Striktни protokol Vremenskog markiranja

If T requests to read x :

- **R1:** If $TS(T) < WTS(x)$, then T is too old; abort T; rollback T
- **R2:** If $TS(T) \geq WTS(x)$, then
 - If the value of x is committed, grant T's read and if $TS(T) > RTS(x)$ assign $TS(T)$ to $RTS(x)$
 - If the value of x is not committed ($C(x)$ is false), T waits (to avoid a dirty read)

If T requests to write x :

- **W1:** If $TS(T) < RTS(x)$, then T is too old; abort T; rollback T
- **W2:** If $RTS(x) \leq TS(T) < WTS(x)$, then no transaction that read x should have read the value T is attempting to write and no transaction will read that value (See R1)
 - If x is committed ($C(x)$ is true), grant the request but do not do the write
 - This is called the Thomas Write Rule
 - If x is not committed, T waits to see if newer value will commit. If it does, discard T's write, else perform it
- **W3:** If $WTS(x), RTS(x) \leq TS(T)$, then
 - If x is committed, grant the request and assign $TS(T)$ to $WTS(x)$, set $C(x)$ to false
 - If the value of x is not committed ($C(x)$ is false), T waits

If T requests to commit:

- **C1:** Set $C(x)$ to true for all elements x written by T, and all transactions that are waiting for x to be committed are allowed to proceed

If T requests to abort:

- **A1:** All T's writes has to be cancelled, and all transactions that are waiting for T, to commit or abort, are allowed to proceed

- Na slici je prikazan redosled izvršavanja skupa transakcija $\{T_1, T_2, T_3\}$
- Upotrebom *striktnog protokola Vremenskog markiranja* proveriti da li se dobija isti redosled. Ako vremenske marke transakcija T_1 , T_2 i T_3 imaju vrednosti 200, 150 i 175, respektivno.

Vreme	T_1	T_2	T_3
t_1	Read(B)		
t_2		Read(A)	
t_3			Read(C)
t_4	Write(B)		
t_5	Write(A)		
t_6	Commit		
t_7		Write(C)	
t_8		Commit	
t_9			Write(A)
t_{10}			Commit

Primer 6 – Protokol Vremenskog markiranja (2)

T	Op	S	RA	WA	CA	RB	WB	CB	RC	WC	CC	T1	T2	T3
			0	0	1	0	0	1	0	0	1	200	150	175
T1	READ (B)	OK				200								
T2	READ (A)	OK	150											
T3	READ (C)	OK							175					
T1	WRITE (B)	OK					200	0						
T1	WRITE(A)	OK		200	0									
T1	COMMIT	OK			1			1						
T2	WRITE (C)	ROLLBACK											225	
T3	WRITE (A)	IGNORE												
T3	COMMIT	OK												
T2	READ (A)	OK	225											
T2	WRITE (C)	OK								225	0			
T2	COMMIT	OK									1			

If T requests to read x:

- **R1:** if $TS(T) < WTS(x)$, then T is too old; abort T; rollback T
- **R2:** if $TS(T) \geq WTS(x)$, then
 - If $C(x)$ is committed ($C(x)$ is true), grant T's read and if $TS(T) > RTS(x)$ assign $TS(T)$ to $RTS(x)$
 - Else T waits

If T requests to write x:

- **W1:** If $TS(T) < RTS(x)$, then T is too old; abort T; rollback T
- **W2:** If $RTS(x) \leq TS(T) < WTS(x)$
 - If $C(x)$ is committed ($C(x)$ is true), grant the request but do not do the write (TWR)
 - Else T waits
- **W3:** If $WTS(x), RTS(x) \leq TS(T)$, then
 - If $C(x)$ is committed, grant the request and assign $TS(T)$ to $WTS(x)$ and set $C(x)$ to false
 - Else T waits

- Na slici je prikazan redosled izvršavanja skupa transakcija $\{T_1, T_2, T_3\}$
- Upotrebom *modifikovanog protokola Vremenskog markiranja* proveriti da li se dobija isti redosled. Ako vremenske marke transakcija T_1 , T_2 i T_3 imaju vrednosti 200, 150 i 175, respektivno.

Vreme	T_1	T_2	T_3
t_1	Read(B)		
t_2		Read(A)	
t_3			Read(C)
t_4	Write(B)		
t_5	Write(A)		
t_6	Commit		
t_7		Write(C)	
t_8		Commit	
t_9			Write(A)
t_{10}			Commit

- $TS(T_2) < TS(T_3) < TS(T_1) \Rightarrow T_2 \rightarrow T_3 \rightarrow T_1$
- t_1 T_1 : Read(B) $TS(T_1) > WTS(B); \Rightarrow$ Read(B)
 $RTS(B) := \text{Max}(RTS(B), TS(T_1)) = 200$
- t_2 T_2 : Read(A) $TS(T_2) > WTS(A); \Rightarrow$ Read(A)
 $RTS(A) := \text{Max}(RTS(A), TS(T_2)) = 150$
- t_3 T_3 : Read(C) $TS(T_3) > WTS(C); \Rightarrow$ Read(C)
 $RTS(C) := \text{Max}(RTS(C), TS(T_3)) = 175$
- t_4 T_1 : Write(B) $TS(T_1) = RTS(B); TS(T_1) > WTS(B); \Rightarrow$ Write(B)
 $WTS(B) := \text{Max}(WTS(B), TS(T_1)) = 200$
- t_5 T_1 : Write(A) $TS(T_1) > RTS(A); TS(T_1) > WTS(A); \Rightarrow$ Write(A)
 $WTS(A) := \text{Max}(WTS(A), TS(T_1)) = 200$
- t_6 T_1 : Commit $(TS(T_2) < TS(T_3) < TS(T_1))^6 =$
 $= (TS(T_2) < TS(T_3) < TS(T_1))^0$
 $TS(T_1)) \neq \text{Min}(TS(T_1), TS(T_2), TS(T_3))^6 \Rightarrow$
 $\text{Set_Commit_Flag}(T_1)$
- t_7 T_2 : Write(C) $TS(T_2) < RTS(A) \Rightarrow \text{Reject}(\text{Write}(C))$
 $\text{Rollback}(T_2); \text{Restart}(T_2);$
 $TS(T_2) := 225 > \text{Max}(TS(T_1), TS(T_2), TS(T_3))$
- Od "scheduler"-a operativnog sistema zavisi sledeća transakcija

- t_8 T_2 : Read(A) $TS(T_2) > WTS(A); \Rightarrow \text{Read}(A)$
 $RTS(A) := \text{Max}(RTS(A), TS(T_2)) = 225$
- t_9 T_3 : Write(A) $TS(T_3) < RTS(A) \Rightarrow \text{Reject}(\text{Write}(A))$
 $\text{Rollback}(T_3); \text{Restart}(T_3);$
 $TS(T_3) := 250 > \text{Max}(TS(T_1), TS(T_2), TS(T_3))$
- t_{10} T_2 : Write(C) $TS(T_2) > RTS(C); TS(T_2) > WTS(C); \Rightarrow \text{Write}(C)$
 $WTS(C) := \text{Max}(WTS(C), TS(T_2)) = 225$
- t_{11} T_2 : Commit $(TS(T_1) < TS(T_2) < TS(T_3))^{11} \neq$
 $\neq (TS(T_2) < TS(T_3) < TS(T_1))^0$
 $\text{Reject}(\text{Commit}); \text{Rollback}(T_2); \text{Restart}(T_2);$
 $TS(T_2) := 275 > \text{Max}(TS(T_1), TS(T_2), TS(T_3))$

 $\text{Commit_Flag}(T_1) \ \& \ (TS(T_1) > TS(T_2))^0 \Rightarrow$
 $\text{Rollback}(T_1); \text{Restart}(T_1);$
 $TS(T_1) := 300 > \text{Max}(TS(T_1), TS(T_2), TS(T_3))$
- t_{12} T_2 : Read(A) $TS(T_2) > WTS(A); \Rightarrow \text{Read}(A)$
 $RTS(A) := \text{Max}(RTS(A), TS(T_2)) = 275$
- t_{13} T_2 : Write(C) $TS(T_2) > RTS(C); TS(T_2) > WTS(C); \Rightarrow \text{Write}(C)$
 $WTS(C) := \text{Max}(WTS(C), TS(T_2)) = 275$

- t_{14} T_3 : Read(C) $TS(T_3) < WTS(C) \Rightarrow \text{Reject}(\text{Read}(C))$
 $\text{Rollback}(T_3); \text{Restart}(T_3);$
 $TS(T_3) := 350 > \text{Max}(TS(T_1), TS(T_2), TS(T_3))$
- t_{15} T_2 : Commit $(TS(T_2) < TS(T_3) < TS(T_1))^{15} =$
 $= (TS(T_2) < TS(T_3) < TS(T_1))^0$
 $TS(T_1)) = \text{Min}(TS(T_1), TS(T_2), TS(T_3))^{15} \Rightarrow$
 $\text{Commit}(T_2)$
- t_{16} T_1 : Read(B) $TS(T_1) > WTS(B); \Rightarrow \text{Read}(B)$
 $RTS(B) := \text{Max}(RTS(B), TS(T_1)) = 300$
- t_{17} T_3 : Write(A) $TS(T_3) > RTS(A); TS(T_3) > WTS(A); \Rightarrow \text{Write}(A)$
 $WTS(A) := \text{Max}(WTS(A), TS(T_3)) = 350$
- t_{18} T_3 : Commit $(TS(T_1) < TS(T_2) < TS(T_3))^{18} \neq$
 $\neq (TS(T_2) < TS(T_3) < TS(T_1))^0$
 $\text{Reject}(\text{Commit}); \text{Rollback}(T_3); \text{Restart}(T_3);$
 $TS(T_3) := 375 > \text{Max}(TS(T_1), TS(T_2), TS(T_3))$
- t_{19} T_1 : Write(B) $TS(T_1) = RTS(B); TS(T_1) > WTS(B); \Rightarrow \text{Write}(B)$
 $WTS(B) := \text{Max}(WTS(B), TS(T_1)) = 300$
- t_{20} T_1 : Write(A) $TS(T_1) < WTS(A); \Rightarrow \text{Ignore}(\text{Write}(A))$

- t_{21} T_1 : Commit $(TS(T_1) < TS(T_3))^{21} \neq (TS(T_3) < TS(T_1))^0$
Reject(Commit); Rollback(T_1); Restart(T_1);
 $TS(T_1) := 400 > \text{Max}(TS(T_1), TS(T_3))$
- t_{22} T_1 : Read(B) $TS(T_1) > WTS(B); \Rightarrow \text{Read}(B)$
 $RTS(B) := \text{Max}(RTS(B), TS(T_1)) = 400$
- t_{23} T_1 : Write(B) $TS(T_1) = RTS(B); TS(T_1) > WTS(B); \Rightarrow \text{Write}(B)$
 $WTS(B) := \text{Max}(WTS(B), TS(T_1)) = 400$
- t_{24} T_3 : Read(C) $TS(T_3) > WTS(C); \Rightarrow \text{Read}(C)$
 $RTS(C) := \text{Max}(RTS(C), TS(T_3)) = 375$
- t_{25} T_3 : Write(A) $TS(T_3) > RTS(A); TS(T_3) > WTS(A); \Rightarrow \text{Write}(A)$
 $WTS(A) := \text{Max}(WTS(A), TS(T_3)) = 375$
- t_{26} T_3 : Commit $(TS(T_3) < TS(T_1))^{26} = (TS(T_3) < TS(T_1))^0$
 $TS(T_3) = \text{Min}(TS(T_1), TS(T_3))^{26} \Rightarrow \text{Commit}(T_3)$
- t_{27} T_1 : Write(A) $TS(T_1) > RTS(A); TS(T_1) > WTS(A); \Rightarrow \text{Write}(A)$
 $WTS(A) := \text{Max}(WTS(A), TS(T_1)) = 400$

- t_{28} T_1 : Commit $(TS(T_3) < TS(T_1))^{28} =$
 $= (TS(T_3) < TS(T_1))^0$
 $TS(T_1)) = \text{Min} (TS(T_1))^{28} \Rightarrow$
 Commit(T_1)
- t_0 $TS(T_1)=200, TS(T_2)=150, TS(T_3)=175 \Rightarrow$
 $TS(T_2) < TS(T_3) < TS(T_1) \Rightarrow T_2 \rightarrow T_3 \rightarrow T_1$
- t_{28} $TS(T_1)=400, TS(T_2)=275, TS(T_3)=375 \Rightarrow$
 $TS(T_2) < TS(T_3) < TS(T_1) \Rightarrow T_2 \rightarrow T_3 \rightarrow T_1$